DICOMautomaton Reference Manual

2021-Aug-15, commit d7361fdb

Contents

1 Overview

1.1 About

DICOMautomaton is a collection of software tools for processing and analyzing medical images. Once a workflow has been developed, the aim of DICOMautomaton is to require minimal interaction to perform the workflow in an automated way. However, some interactive tools are also included for workflow development, exploratory analysis, and contouring.

DICOMautomaton is meant to be flexible enough to adapt to a wide variety of situations and has been incorporated into projects to provide: a local PACs, image analysis for various types of QA, kinetic modeling of perfusion images, automated fuzzy mapping of ROI names to a standard lexicon, dosimetric analysis, TCP and NTCP modeling, ROI contour/volume manipulation, estimation of surface dose, ray casting through patient and phantom geometry, rudimentary linac beam optimization, radiomics, and has been used in various ways to explore the relationship between toxicity and dose in sub-organ compartments.

Note: DICOMautomaton should **NOT** be used for clinical purposes. It is experimental software. It is suitable for research or support tool purposes only.

It comes with no warranty or guarantee of any kind, either explicit or implied. Users of DICOMautomaton do so fully at their own risk.

1.2 Project Home

This project's homepage can be found at http://www.halclark.ca/. The source code is available at either https://gitlab.com/hdeanclark/DICOMautomaton/ or https://github.com/hdclark/DICOMautomaton/.

1.3 Download

DICOMautomaton relies only on open source software and is itself open source software. Source code is available at https://github.com/hdclark/DICOMautomaton.

Currently, binaries are not provided. Only linux is supported and a recent C++ compiler is needed. A `PKGBUILD` file is provided for Arch Linux and derivatives, and CMake can be used to generate deb files for Debian derivatives. A docker container is available for easy portability to other systems. DICOMautomaton has successfully run on x86, x86_64, and most ARM systems. To maintain flexibility, DICOMautomaton is generally not ABI or API stable.

1.4 License and Copying

1.5 Dependencies

Dependencies are listed in the `PKGBUILD` file (using Arch Linux package naming conventions) and in the `CMakeLists.txt` file (Debian package naming conventions) bundled with the source code. See https://github.com/hdclark/DICOMautomaton. Broadly, DICOMautomaton depends on Boost, CGAL, SFML, SDL2, glew, Eigen, Asio, Wt, NLopt, and PostgreSQL. Disabling some functionality at

compile time can eliminate some dependencies. This instance has been compiled with the following functionality.

Table 1: Dependencies enabled for this instance.

Dependency	Functionality Enabled?
Ygor	true (required)
YgorClustering	true (required; header-only)
Explicator	true (required)
Imebra	true (required; bundled)
Boost	true (required)
asio	true (required)
zlib	true (required)
MPFR	true (required)
GNU GMP	true (required)
Eigen	true
CGAL	true
NLOpt	true
SFML	true
SDL2	true
glew	true
Wt	true
GNU GSL	true
pqxx	true
Jansson	true

Notably, DICOMautomaton depends on the author's 'Ygor,' 'Explicator,' and 'YgorClustering' projects. See https://gitlab.com/hdeanclark/Ygor (mirrored at https://github.com/hdclark/Ygor), https://gitlab.com/hdeanclark/Explicator (mirrored at https://github.com/hdclark/Explicator), and (only for compilation) https://gitlab.com/hdeanclark/YgorClustering (mirrored at https://github.com/hdclark/YgorClustering).

1.6 Feedback

All feedback, questions, comments, and pull requests are welcomed.

1.7 FAQs

Q. What is the best way to use DICOMautomaton?
A. DICOMautomaton provides a command-line interface, SFML-based image viewer, and limited web interface. The command-line interface is most conducive to automation, the viewer works best for interactive tasks, and the web interface works well for specific installations.

Q. How do I contribute, report bugs, or contact the author?
A. All feedback, questions, comments, and pull requests are welcomed. Please find contact information at https://github.com/hdclark/DICOMautomaton.

1.8 Citing

DICOMautomaton can be cited as a whole using doi:10.5281/zenodo.4088796. Individual releases are assigned a DOI too; the latest release DOI can be found here.

Several publications and presentations refer to DICOMautomaton or describe some aspect of it. Here are a few:

- H. Clark, J. Beaudry, J. Wu, and S. Thomas. Making use of virtual dimensions for visualization and contouring. Poster presentation at the International Conference on the use of Computers in Radiation Therapy, London, UK. June 27-30,

  2016.

- H. Clark, S. Thomas, V. Moiseenko, R. Lee, B. Gill, C. Duzenli, and J. Wu. Automated segmentation and dose-volume analysis with DICOMautomaton. In the Journal of Physics: Conference Series, vol. 489, no. 1, p. 012009. IOP Publishing, 2014.

- H. Clark, J. Wu, V. Moiseenko, R. Lee, B. Gill, C. Duzenli, and S. Thomas. Semi-automated contour recognition using DICOMautomaton. In the Journal of Physics: Conference Series, vol. 489, no. 1, p. 012088. IOP Publishing, 2014.

- H. Clark, J. Wu, V. Moiseenko, and S. Thomas. Distributed, asynchronous, reactive dosimetric and outcomes analysis using DICOMautomaton. Poster presentation at the COMP Annual Scientific Meeting, Banff, Canada. July 9-12, 2014.

If you use DICOMautomaton in an academic work, we ask that you please cite the most relevant publication for that work or the most relevant release DOI, if possible.

1.9 Components

1.9.1 dicomautomaton_dispatcher

1.9.1.1 Description The core command-line interface to DICOMautomaton is the `dicomautomaton_dispatcher` program. It is presents an interface based on chaining of discrete operations on collections of images, DICOM images, DICOM radiotherapy files (RTSTRUCTS and RTDOSE), and various other types of files. `dicomautomaton_dispatcher` has access to all defined operations

described in Operations. It can be used to launch both interactive and non-interactive tasks. Data can be sourced from a database or files in a variety of formats.

Name/label selectors in dicomautomaton_dispatcher generally support fuzzy matching via libexplicator or regular expressions. The operations and parameters that provide these options are documented in Operations.

Filetype support differs in some cases. A custom FITS file reader and writer are supported, and DICOM files are generally supported. There is currently no support for RTPLANs, though DICOM image, RTSTRUCT, and RTDOSE files are well supported. There is limited support for writing files – currently JPEG, PNG, and FITS images; RTDOSE files; and Boost.Serialize archive writing are supported.

1.9.1.2 Usage Examples

- `dicomautomaton_dispatcher --help`
  Print a listing of all available options.
- `dicomautomaton_dispatcher CT*dcm`
  Launch the default interactive viewer to inspect a collection of computed tomography images.
- `dicomautomaton_dispatcher MR*dcm`
  Launch the default interactive viewer to inspect a collection of magnetic resonance images.
- `dicomautomaton_dispatcher -o SFML_Viewer MR*dcm`
  *Launch the default interactive viewer to inspect a collection of magnetic resonance images. Note that files specified on the command line are always loaded **prior** to running any operations. Injecting files midway through the operation chain must make use of an operation designed to do so.*
- `dicomautomaton_dispatcher CT*dcm RTSTRUCT*dcm RTDOSE*dcm -o Average -o SFML_Viewer`
  Load DICOM files, perform an averaging operation, and then launch the SFML viewer to inspect the output.
- `dicomautomaton_dispatcher ./RTIMAGE.dcm -o AnalyzePicketFence:ImageSelection='last':Int`
  Perform a picket fence analysis of an RTIMAGE file.
- `dicomautomaton_dispatcher -f create_temp_view.sql -f select_records_from_temp_view.sql -o ComputeSomething`
  Load a SQL common file that creates a SQL view, issue a query involving the view which returns some DICOM file(s). Perform analysis 'ComputeSomething' with the files.
- `dicomautomaton_dispatcher -f common.sql -f seriesA.sql -n -f seriesB.sql -o SFML_Viewer`

Load two distinct groups of data. The second group does not 'see' the file 'common.sql' side effects – the queries are totally separate.

- `dicomautomaton_dispatcher fileA fileB -s fileC adir/ -m PatientID=XYZ003 -o ComputeXYZ -o SFML_Viewer`
  Load standalone files and all files in specified directory. Inform the analysis 'ComputeXYZ' of the patient's ID, launch the analysis, and then interactively view.

- `dicomautomaton_dispatcher CT*dcm -o ModifyingOperation -o BoostSerializeDrover`
  Launch the default interactive viewer to inspect a collection of computed tomography images, perform an operation that modifies them, and serialize the internal state for later using the BoostSerializeDrover operation.

1.9.2 dicomautomaton_webserver

1.9.2.1 Description This web server presents most operations in an interactive web page. Some operations are disabled by default (e.g., BuildLexiconInteractively because they are not designed to be operated via remote procedure calls. This routine should be run within a capability-limiting environment, but access to an X server is required. A Docker script is bundled with DICOMautomaton sources which includes everything needed to function properly.

1.9.2.2 Usage Examples

- `dicomautomaton_webserver --help`
  Print a listing of all available options. Note that most configuration is done via editing configuration files. See `/etc/DICOMautomaton/`.

- `dicomautomaton_webserver --config /etc/DICOMautomaton/webserver.conf --http-address 0.0.0.0 --http-port 8080   --docroot='/etc/DICOMautomaton/'`
  Launch the webserver on any interface and port 8080.

1.9.3 dicomautomaton_bsarchive_convert

1.9.3.1 Description A program for converting Boost.Serialization archives types which DICOMautomaton can read. These archives need to be created by the BoostSerializeDrover operation. Some archive types are concise and not portable (i.e., binary archives), or verbose (and thus slow to read and write) and portable (i.e., XML, plain text). To combat verbosity, on-the-fly gzip compression and decompression is supported. This program can be used to convert archive types.

1.9.3.2 Usage Examples

- `dicomautomaton_bsarchive_convert --help`
  Print a listing of all available options.

- `dicomautomaton_bsarchive_convert -i file.binary -o file.xml -t 'XML'`
  Convert a binary archive to a portable XML archive.
- `dicomautomaton_bsarchive_convert -i file.binary.gz -o file.xml.gz -t 'gzip-xml'`
  Convert a binary archive to a gzipped portable XML archive.
- `dicomautomaton_bsarchive_convert -i file.binary.gz -o file.xml -t 'XML'`
  Convert a gzipped binary archive to a non-gzipped portable XML archive.
- `dicomautomaton_bsarchive_convert -i file.xml.gz -o file.txt -t 'txt'`
  Convert a gzipped binary archive to a non-gzipped, portable, and inspectable text archive.
- `dicomautomaton_bsarchive_convert -i file.txt -o file.txt.gz -t 'gzip-txt'`
  Convert an uncompressed text archive to a compressed text archive. Note that this conversion is effectively the same as simply ***`gzip file.txt`****.*
- `dicomautomaton_bsarchive_convert -i file.xml.gz -o file.bin -t 'binary'`
  Convert a compressed archive to a binary file. Note that binary archives should only expect to be readable on the same hardware with the same versions and are therefore best for checkpointing calculations that can fail or may need to be tweaked later.*
- `dicomautomaton_bsarchive_convert -i file.xml.gz -o file.bin.gz -t 'gzip-binary'`
  Convert a compressed archive to a compressed binary file.

1.9.4 dicomautomaton_dump

1.9.4.1 Description This program is extremely simplistic. Given a single DICOM file, it prints to stdout the value of one DICOM tag. This program is best used in scripts, for example to check the modality or a file.

1.9.4.2 Usage Examples

- `dicomautomaton_dump afile.dcm 0x0008 0x0060`
  Print the value of the DICOM tag (0x0008,0x0060) aka (0008,0060).

1.9.5 pacs_ingress

1.9.5.1 Description Given a DICOM file and some additional metadata, insert the data into a PACs system database. The file itself will be copied into the database and various bits of data will be deciphered. Note that at the moment a 'gdcmdump' file must be provided and is stored alongside the DICOM

file in the database filestore. This sidecar file is meant to support ad-hoc DICOM queries without having to index the entire file. Also note that imports into the database are minimal, leaving files with multiple NULL values. This is done to improve ingress times. A separate database refresh (pacs_refresh) must be performed to replace NULL values.

1.9.5.2 Usage Examples

- `pacs_ingress --help`
  Print a listing of all available options.
- `pacs_ingress -f '/tmp/a.dcm' -g '/tmp/a.gdcmdump' -p 'XYZ Study 2019' -c 'Study concerning XYZ.'`
  Insert the file '/tmp/a.dcm' into the database.

1.9.6 pacs_refresh

1.9.6.1 Description A program for trying to replace database NULLs, if possible, using stored files. This program is complementary to pacs_ingress. Note that the `--days-back/-d` parameter should always be specified.

1.9.6.2 Usage Examples

- `pacs_refresh --help`
  Print a listing of all available options.
- `pacs_refresh -d 7`
  Perform a refresh of the database, restricting to files imported within the previous 7 days.

1.9.7 pacs_duplicate_cleaner

1.9.7.1 Description Given a DICOM file, check if it is in the PACS DB. If so, delete the file. Note that a full, byte-by-byte comparison is NOT performed – rather only the top-level DICOM unique identifiers are (currently) compared. No other metadata is considered. So this program is not suitable if DICOM files have been modified without re-assigning unique identifiers! (Which is non-standard behaviour.) Note that if an *exact* comparison is desired, using a traditional file de-duplicator will work.

1.9.7.2 Usage Examples

- `pacs_duplicate_cleaner --help`
  Print a listing of all available options.
- `pacs_duplicate_cleaner -f '/path/to/a/dicom/file.dcm'`
  Check if 'file.dcm' is already in the PACS DB. If so, delete it ('file.dcm').

- `pacs_duplicate_cleaner -f '/path/to/a/dicom/file.dcm' -n`
  Check if 'file.dcm' is already in the PACS DB, but do not delete anything.

2 List of Available Operations

- AccumulateRowsColumns
- AnalyzeHistograms
- AnalyzeLightRadFieldCoincidence
- AnalyzePicketFence
- AnalyzeTPlan
- ApplyCalibrationCurve
- AutoCropImages
- Average
- BCCAExtractRadiomicFeatures
- BEDConvert
- BoostSerializeDrover
- BuildLexiconInteractively
- CT_Liver_Perfusion
- CT_Liver_Perfusion_First_Run
- CT_Liver_Perfusion_Ortho_Views
- CT_Liver_Perfusion_Pharmaco_1C2I_5Param
- CT_Liver_Perfusion_Pharmaco_1C2I_Reduced3Param
- CellularAutomata
- ClusterDBSCAN
- ComparePixels
- ContourBasedRayCastDoseAccumulate
- ContourBooleanOperations
- ContourSimilarity
- ContourViaGeometry
- ContourViaThreshold
- ContourVote
- ContourWholeImages
- ContouringAides
- ConvertContoursToImages (alias for HighlightROIs)
- ConvertContoursToMeshes
- ConvertContoursToPoints
- ConvertDoseToImage
- ConvertImageToDose
- ConvertImageToMeshes
- ConvertImagesToContours (alias for ContourViaThreshold)
- ConvertMeshesToContours
- ConvertMeshesToPoints
- ConvertNaNsToAir
- ConvertNaNsToZeros
- ConvertPixelsToPoints

- ConvolveImages
- CopyContours
- CopyImages
- CopyMeshes
- CopyPoints
- CountVoxels
- CropImageDoseToROIs
- CropImages
- CropROIDose
- DCEMRI_IAUC
- DCEMRI_Nonparametric_CE
- DICOMExportContours
- DICOMExportImagesAsCT
- DICOMExportImagesAsDose
- DeDuplicateImages
- DecayDoseOverTimeHalve
- DecayDoseOverTimeJones2014
- DecimatePixels
- DeleteContours
- DeleteImages
- DeleteMeshes
- DeletePoints
- DetectGrid3D
- DetectShapes3D
- DrawGeometry
- DroverDebug
- DumpAllOrderedImageMetadataToFile
- DumpAnEncompassedPoint
- DumpFilesPartitionedByTime
- DumpImageMeshes
- DumpImageMetadataOccurrencesToFile
- DumpPerROIParams_KineticModel_1C2I_5P
- DumpPixelValuesOverTimeForAnEncompassedPoint
- DumpPlanSummary
- DumpROIContours
- DumpROIData
- DumpROISNR
- DumpROISurfaceMeshes
- DumpTPlanMetadataOccurrencesToFile
- DumpVoxelDoseInfo
- EvaluateDoseVolumeStats
- EvaluateNTCPModels
- EvaluateTCPModels
- ExportFITSImages
- ExportLineSamples
- ExportPointClouds

- ExportSurfaceMeshes
- ExportSurfaceMeshesOBJ
- ExportSurfaceMeshesOFF
- ExportSurfaceMeshesPLY
- ExportSurfaceMeshesSTL
- ExportWarps
- ExtractAlphaBeta
- ExtractImageHistograms
- ExtractPointsWarp
- ExtractRadiomicFeatures
- FVPicketFence
- ForEachDistinct
- GenerateCalibrationCurve
- GenerateSurfaceMask
- GenerateSyntheticImages
- GenerateVirtualDataContourViaThresholdTestV1
- GenerateVirtualDataDoseStairsV1
- GenerateVirtualDataImageSphereV1
- GenerateVirtualDataPerfusionV1
- GenerateWarp
- GiveWholeImageArrayABoneWindowLevel
- GiveWholeImageArrayAHeadAndNeckWindowLevel
- GiveWholeImageArrayAThoraxWindowLevel
- GiveWholeImageArrayAnAbdominalWindowLevel
- GiveWholeImageArrayAnAlphaBetaWindowLevel
- GridBasedRayCastDoseAccumulate
- GroupImages
- GrowContours
- HighlightROIs
- ImageRoutineTests
- ImprintImages
- InterpolateSlices
- IsolatedVoxelFilter
- LoadFiles
- LogScale
- MakeMeshesManifold
- MaxMinPixels
- MeldDose
- MinkowskiSum3D
- ModelIVIM
- ModifyContourMetadata
- ModifyImageMetadata
- NegatePixels
- NoOp
- NormalizeLineSamples
- NormalizePixels

- OptimizeStaticBeams
- OrderImages
- PartitionContours
- PartitionImages (alias for GroupImages)
- PlotLineSamples
- PlotPerROITimeCourses
- PointSeparation
- PreFilterEnormousCTValues
- PresentationImage
- PruneEmptyImageDoseArrays
- PurgeContours
- RankPixels
- ReduceNeighbourhood
- RemeshSurfaceMeshes
- Repeat
- SDL_Viewer
- SFML_Viewer
- ScalePixels
- SeamContours
- SelectSlicesIntersectingROI
- SimplifyContours
- SimplifySurfaceMeshes
- SimulateRadiograph
- SpatialBlur
- SpatialDerivative
- SpatialSharpen
- SubdivideSurfaceMeshes
- SubsegmentContours
- Subsegment_ComputeDose_VanLuijk
- SubtractImages
- SupersampleImageGrid
- SurfaceBasedRayCastDoseAccumulate
- ThresholdImages
- ThresholdOtsu
- TrimROIDose
- UBC3TMRI_DCE
- UBC3TMRI_DCE_Differences
- UBC3TMRI_DCE_Experimental
- UBC3TMRI_IVIM_ADC
- VolumetricCorrelationDetector
- VolumetricSpatialBlur
- VolumetricSpatialDerivative
- VoxelRANSAC
- WarpContours
- WarpImages
- WarpMeshes

- WarpPoints

3 Operations Reference

3.1 AccumulateRowsColumns

3.1.1 Description

This operation generates row- and column-profiles of images in which the entire row or column has been summed together. It is useful primarily for detection of axes-aligned edges or ridges.

3.1.2 Notes

- It is often useful to pre-process inputs by computing an in-image-plane derivative, gradient magnitude, or similar (i.e., something to emphasize edges) before calling this routine. It is not necessary, however.

3.1.3 Parameters

- ImageSelection

3.1.3.1 ImageSelection

3.1.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.1.3.1.2 Default

- `"last"`

3.1.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.2 AnalyzeHistograms

3.2.1 Description

This operation analyzes the selected line samples as if they were cumulative dose-volume histograms (DVHs). Multiple criteria can be specified. The output is a CSV file that can be concatenated or appended to other output files to provide a summary of multiple criteria.

3.2.2 Notes

- This routine will filter out non-matching line samples. Currently required: Modality=Histogram; each must be explicitly marked as a cumulative, unscaled abscissa + unscaled ordinate histogram; and differential distribution statistics must be available (e.g., min, mean, and max voxel doses).

3.2.3 Parameters

- LineSelection
- SummaryFilename
- UserComment
- Description
- Constraints
- ReferenceDose

3.2.3.1 LineSelection

3.2.3.1.1 Description Select one or more line samples. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth line sample (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last line sample. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the line sample composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all line sample that do not have the greatest number of sub-objects, not the least-numerous line sample (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.2.3.1.2 Default

- `"last"`

3.2.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.2.3.2 SummaryFilename

3.2.3.2.1 Description A summary of the criteria and results will be appended to this file. The format is CSV. Leave empty to dump to generate

a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.2.3.2.2 Default

- ""

3.2.3.2.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.2.3.3 UserComment

3.2.3.3.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. Even if left empty, the column will remain in the output to ensure the outputs from multiple runs can be safely concatenated. Preceding alphanumeric variables with a '$' will cause them to be treated as metadata keys and replaced with the corresponding key's value, if present. For example, 'The modality is $Modality' might be (depending on the metadata) expanded to 'The modality is Histogram'. If the metadata key is not

3.2.3.3.2 Default

- ""

3.2.3.3.3 Examples

- "Using XYZ"
- "Patient treatment plan C"
- "$PatientID"

3.2.3.4 Description

3.2.3.4.1 Description A string that will be inserted into the output file which should be used to describe the constraint and any caveats that the viewer should be aware of. Generally, the UserComment is best for broadly-defined notes whereas the Description is tailored for each constraint. Preceding alphanumeric variables with a '$' will cause them to be treated as metadata keys and replaced with the corresponding key's value, if present. For example, 'The modality is $Modality' might be (depending on the metadata) expanded to 'The modality is Histogram'. If the metadata key is not

3.2.3.4.2 Default

- ""

3.2.3.4.3 Examples

- "Liver"
- "Lung"
- "Liver - GTV"
- "$LineName"

3.2.3.5 Constraints

3.2.3.5.1 Description Constraint criteria that will be evaluated against the selected line samples. There three general types of constraints will be recognized. First, constraints in the style of 'Dmax < 50.0 Gy'. The left-hand-size (LHS) can be any of {Dmin, Dmean, Dmax}. The inequality can be any of {<, lt, <=, lte, >, gt, >=, gte}. The right-hand-side (RHS) units can be any of {Gy, %} where '%' means the RHS number is a percentage of the ReferenceDose. Second, constraints in the style of 'D(coldest 500.0 cc) < 50.4 Gy'. The inner LHS can be any of {coldest, hottest}. The inner LHS units can be any of {cc, cm3, cm^3, %} where '%' means the inner LHS number is a percentage of the total volume. The inequality can be any of {<, lt, <=, lte, >, gt, >=, gte}. The RHS units can be any of {Gy, %} where '%' means the RHS number is a percentage of the ReferenceDose. Third, constraints in the style of 'V(24.5 Gy) < 500.0 cc'. The inner LHS units can be any of {Gy, %} where '%' means the inner LHS number is a percentage of the ReferenceDose. The inequality can be any of {<, lt, <=, lte, >, gt, >=, gte}. The RHS units can be any of {cc, cm3, cm^3, %} where '%' means the inner LHS number is a percentage of the total volume. Multiple constraints can be supplied by separating them with ';' delimiters. Each will be evaluated separately. Newlines can also be used, though constraints should all end with a ';'. Comments can be included by preceeding with a '#', which facilitate supplying lists of constraints piped in (e.g., from a file via Bash process substitution).

3.2.3.5.2 Default

- ""

3.2.3.5.3 Examples

- "Dmax < 50.0 Gy"
- "Dmean lte 80 %"
- "Dmin >= 80 %"
- "Dmin >= 65 Gy"
- "D(coldest 500.0 cc) <= 25.0 Gy"
- "D(coldest 500.0 cc) <= 15.0 %"

- "D(coldest 50%) <= 15.0 %"
- "D(hottest 10%) gte 95.0 %"
- "V(24.5 Gy) < 500.0 cc"
- "V(10%) < 50.0 cc"
- "V(24.5 Gy) < 500.0 cc"
- "Dmax < 50.0 Gy ; Dmean lte 80 % ; D(hottest 10%) gte 95.0 %"

3.2.3.6 ReferenceDose

3.2.3.6.1 Description The absolute dose that relative (i.e., percentage) constraint doses will be considered against. Generally this will be the prescription dose (in DICOM units; Gy). If there are multiple prescriptions, either the prescription appropriate for the constraint should be supplied, or relative dose constraints should not be used.

3.2.3.6.2 Default

- "nan"

3.2.3.6.3 Examples

- "70.0"
- "42.5"

3.3 AnalyzeLightRadFieldCoincidence

3.3.1 Description

This operation analyzes the selected images to compare light and radiation field coincidence for fixed, symmetric field sizes. Coincidences are extracted automatically by fitting Gaussians to the peak nearest to one of the specified field boundaries and comparing offset from one another. So, for example, a 10x10cm MLC-defined field would be compared to a 15x15cm field if there are sharp edges (say, metal rulers) that define a 10x10cm field (i.e., considered to represent the light field). Horizontal and vertical directions (both positive and negative) are all analyzed separately.

3.3.2 Notes

- This routine assumes both fields are squarely aligned with the image axes. Alignment need not be perfect, but the Gaussians may be significantly broadened if there is misalignment. This should be fixed in a future revision.

- It is often useful to pre-process inputs by computing an in-image-plane derivative, gradient magnitude, or similar (i.e., something to emphasize edges) before calling this routine. It may not be necessary, however.

3.3.3 Parameters

- ImageSelection
- ToleranceLevel
- EdgeLengths
- SearchDistance
- PeakSimilarityThreshold
- UserComment
- OutputFileName
- InteractivePlots

3.3.3.1 ImageSelection

3.3.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.3.3.1.2 Default

- `"last"`

3.3.3.1.3 Examples

- "last"
- "first"
- "all"
- "none"
- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.3.3.2 ToleranceLevel

3.3.3.2.1 Description Controls detected edge visualization for easy identification of edges out of tolerance. Note: this value refers to edge-to-edge separation, not edge-to-nominal distances. This value is in DICOM units.

3.3.3.2.2 Default

- "1.0"

3.3.3.2.3 Examples

- "0.5"
- "1.0"
- "2.0"
- "inf"

3.3.3.3 EdgeLengths

3.3.3.3.1 Description Comma-separated list of (symmetric) edge lengths fields should be analyzed at. For example, if 50x50, 100x100, 150x150, and 200x200 (all in mm) fields are to be analyzed, this argument would be '50,100,150,200' and it will be assumed that the field centre is at DICOM position (0,0,0). All values are in DICOM units.

3.3.3.3.2 Default

- "100"

3.3.3.3.3 Examples

- "100.0"
- "50,100,150,200,300"

- `"10.273,20.2456"`

3.3.3.4 SearchDistance

3.3.3.4.1 Description The distance around the anticipated field edges to search for edges (actually sharp peaks arising from edges). If an edge is further away than this value from the anticipated field edge, then the coincidence will be ignored altogether. The value should be greater than the largest action/tolerance threshold with some additional margin (so gross errors can be observed), but small enough that spurious edges (i.e., unintended features in the image, such as metal fasteners, or artifacts near the field edge) do not replace the true field edges. The 'sharpness' of the field edge (resulting from the density of the material used to demarcate the edge) can impact this value; if the edge is not sharp, then the peak will be shallow, noisy, and may therefore travel around depending on how the image is pre-processed. Note that both radiation field and light field edges may differ from the 'nominal' anticipated edges, so this wobble factor should be incorporated in the search distance. This quantity must be in DICOM units.

3.3.3.4.2 Default

- `"3.0"`

3.3.3.4.3 Examples

- `"2.5"`
- `"3.0"`
- `"5.0"`

3.3.3.5 PeakSimilarityThreshold

3.3.3.5.1 Description Images can be taken such that duplicate peaks will occur, such as when field sizes are re-used. Peaks are therefore de-duplicated. This value (as a %, ranging from [0,100]) specifies the threshold of disimilarity below which peaks are considered duplicates. A low value will make duplicates confuse the analysis, but a high value may cause legitimate peaks to be discarded depending on the attenuation cababilties of the field edge markers.

3.3.3.5.2 Default

- `"25"`

3.3.3.5.3 Examples

- `"5"`
- `"10"`
- `"15"`
- `"50"`

3.3.3.6 UserComment

3.3.3.6.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. If left empty, the column will be omitted from the output.

3.3.3.6.2 Default

- `""`

3.3.3.6.3 Examples

- `""`
- `"6MV"`
- `"Using XYZ"`
- `"Test with thick metal edges"`

3.3.3.7 OutputFileName

3.3.3.7.1 Description A filename (or full path) in which to append field edge coincidence data generated by this routine. The format is CSV. Leave empty to dump to generate a unique temporary file.

3.3.3.7.2 Default

- `""`

3.3.3.7.3 Examples

- `""`
- `"/tmp/somefile"`
- `"localfile.csv"`
- `"derivative_data.csv"`

3.3.3.8 InteractivePlots

3.3.3.8.1 Description Whether to interactively show plots showing detected edges.

3.3.3.8.2 Default

- `"false"`

3.3.3.8.3 Examples

- `"true"`
- `"false"`

3.4 AnalyzePicketFence

3.4.1 Description

This operation extracts MLC positions from a picket fence image.

3.4.2 Notes

- This routine requires data to be pre-processed. The gross picket area should be isolated and the leaf junction areas contoured (one contour per junction). Both can be accomplished via thresholding. Additionally, stray pixels should be filtered out using, for example, median or conservative filters.
- This routine analyzes the picket fences on the plane in which they are specified within the DICOM file, which often coincides with the image receptor ('RTImageSID'). Tolerances are evaluated on the isoplane, so the image is projected before measuring distances, but the image itself is not altered; a uniform magnification factor of SAD/SID is applied to all distances.

3.4.3 Parameters

- ImageSelection
- MLCModel
- MLCROILabel
- JunctionROILabel
- PeakROILabel
- MinimumJunctionSeparation
- ThresholdDistance
- LeafGapsFileName
- ResultsSummaryFileName
- UserComment
- InteractivePlots

3.4.3.1 ImageSelection

3.4.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is

possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.4.3.1.2 Default

- `"last"`

3.4.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.4.3.2 MLCModel

3.4.3.2.1 Description The MLC design geometry to use. 'VarianMillenniumMLC80' has 40 leafs in each bank; leaves are 10mm wide at isocentre; and the maximum static field size is 40cm x 40cm. 'VarianMillenniumMLC120' has 60 leafs in each bank; the 40 central leaves are 5mm wide at isocentre; the 20 peripheral leaves are 10mm wide; and the maximum static field size is 40cm x

40cm. 'VarianHD120' has 60 leafs in each bank; the 32 central leaves are 2.5mm wide at isocentre; the 28 peripheral leaves are 5mm wide; and the maximum static field size is 40cm x 22cm.

3.4.3.2.2 Default

- `"VarianMillenniumMLC120"`

3.4.3.2.3 Supported Options

- `"VarianMillenniumMLC80"`
- `"VarianMillenniumMLC120"`
- `"VarianHD120"`

3.4.3.3 MLCROILabel

3.4.3.3.1 Description An ROI imitating the MLC axes of leaf pairs is created. This is the label to apply to it. Note that the leaves are modeled with thin contour rectangles of virtually zero area. Also note that the outline colour is significant and denotes leaf pair pass/fail.

3.4.3.3.2 Default

- `"Leaves"`

3.4.3.3.3 Examples

- `"MLC_leaves"`
- `"MLC"`
- `"approx_leaf_axes"`

3.4.3.4 JunctionROILabel

3.4.3.4.1 Description An ROI imitating the junction is created. This is the label to apply to it. Note that the junctions are modeled with thin contour rectangles of virtually zero area.

3.4.3.4.2 Default

- `"Junction"`

3.4.3.4.3 Examples

- `"Junction"`
- `"Picket_Fence_Junction"`

3.4.3.5 PeakROILabel

3.4.3.5.1 Description ROIs encircling the leaf profile peaks are created. This is the label to apply to it. Note that the peaks are modeled with small squares.

3.4.3.5.2 Default

- `"Peak"`

3.4.3.5.3 Examples

- `"Peak"`
- `"Picket_Fence_Peak"`

3.4.3.6 MinimumJunctionSeparation

3.4.3.6.1 Description The minimum distance between junctions on the SAD isoplane in DICOM units (mm). This number is used to de-duplicate automatically detected junctions. Analysis results should not be sensitive to the specific value.

3.4.3.6.2 Default

- `"10.0"`

3.4.3.6.3 Examples

- `"5.0"`
- `"10.0"`
- `"15.0"`
- `"25.0"`

3.4.3.7 ThresholdDistance

3.4.3.7.1 Description The threshold distance in DICOM units (mm) above which MLC separations are considered to 'fail'. Each leaf pair is evaluated separately. Pass/fail status is also indicated by setting the leaf axis contour colour (blue for pass, red for fail).

3.4.3.7.2 Default

- `"1.0"`

3.4.3.7.3 Examples

- `"0.5"`
- `"1.0"`
- `"2.0"`

3.4.3.8 LeafGapsFileName

3.4.3.8.1 Description This file will contain gap and nominal-vs-actual offset distances for each leaf pair. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.4.3.8.2 Default

- ""

3.4.3.8.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.4.3.9 ResultsSummaryFileName

3.4.3.9.1 Description This file will contain a brief summary of the results. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.4.3.9.2 Default

- ""

3.4.3.9.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.4.3.10 UserComment

3.4.3.10.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data.

3.4.3.10.2 Default

- ""

3.4.3.10.3 Examples

- `""`
- `"Using XYZ"`
- `"Patient treatment plan C"`

3.4.3.11 InteractivePlots

3.4.3.11.1 Description Whether to interactively show plots showing detected edges.

3.4.3.11.2 Default

- `"false"`

3.4.3.11.3 Examples

- `"true"`
- `"false"`

3.5 AnalyzeTPlan

3.5.1 Description

This operation analyzes the selected RT plans, performing a general analysis suitable for exploring or comparing plans at a high-level. Currently, only the total leaf opening (i.e., the sum of all leaf openings – the distance between a leaf in bank A to the opposing leaf in bank B) is reported for each plan, beam, and control point. The output is a CSV file that can be concatenated or appended to other output files to provide a summary of multiple criteria.

3.5.2 Parameters

- TPlanSelection
- SummaryFilename
- UserComment
- Description

3.5.2.1 TPlanSelection

3.5.2.1.1 Description Select one or more treatment plans. Note that a single treatment plan may be composed of multiple beams; if delivered sequentially, they should collectively represent a single logically cohesive plan. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be ‘first’, ‘last’, ‘none’, or ‘all’ literals. Additionally ‘#N’ for some positive integer N selects the Nth treatment plan (with zero-based indexing). Likewise, ‘#-N’ selects the Nth-from-last treatment plan. Positional specifiers can be inverted by prefixing with a ‘!’.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a ‘!’). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the ‘numerous’ and ‘fewest’ literals, which selects the treatment plan composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a ‘!’. Note that ‘!numerous’ means all treatment plan that do not have the greatest number of sub-objects, not the least-numerous treatment plan (i.e., ‘fewest’).

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ‘;’ and are applied in the order specified.

3.5.2.1.2 Default

- `"last"`

3.5.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.5.2.2 SummaryFilename

3.5.2.2.1 Description Analysis results will be appended to this file. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.5.2.2.2 Default

- `""`

3.5.2.2.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.5.2.3 UserComment

3.5.2.3.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. Even if left empty, the column will remain in the output to ensure the outputs from multiple runs can be safely concatenated. Preceding alphanumeric variables with a '$' will cause them to be treated as metadata keys and replaced with the corresponding key's value, if present. For example, 'The modality is $Modality' might be (depending on the metadata) expanded to 'The modality is RTPLAN'. If the metadata key is notp

3.5.2.3.2 Default

- ""

3.5.2.3.3 Examples

- "Using XYZ"
- "Patient treatment plan C"
- "$PatientID"

3.5.2.4 Description

3.5.2.4.1 Description A string that will be inserted into the output file which should be used to describe the constraint and any caveats that the viewer should be aware of. Generally, the UserComment is best for broadly-defined notes whereas the Description is tailored for each constraint. Preceding alphanumeric variables with a '$' will cause them to be treated as metadata keys and replaced with the corresponding key's value, if present. For example, 'The modality is $Modality' might be (depending on the metadata) expanded to 'The modality is RTPLAN'. If the metadata key is notp

3.5.2.4.2 Default

- ""

3.5.2.4.3 Examples

- "2 Arcs"
- "1 Arc"
- "IMRT"

3.6 ApplyCalibrationCurve

3.6.1 Description

This operation applies a given calibration curve to voxel data inside the specified ROI(s). It is designed to apply calibration curves, but is useful for transforming voxel intensities using any supplied 1D curve.

3.6.2 Notes

- This routine can handle overlapping or duplicate contours.

3.6.3 Parameters

- Channel
- ImageSelection
- ContourOverlap
- Inclusivity
- CalibCurveFileName
- NormalizedROILabelRegex
- ROILabelRegex

3.6.3.1 Channel

3.6.3.1.1 Description The image channel to use. Zero-based. Use '-1' to operate on all available channels.

3.6.3.1.2 Default

- `"-1"`

3.6.3.1.3 Examples

- `"-1"`
- `"0"`
- `"1"`
- `"2"`

3.6.3.2 ImageSelection

3.6.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be ‘first’, ‘last’, ‘none’, or ‘all’ literals. Additionally ‘#N’ for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, ‘#-N’ selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a ‘!’.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a ‘!’). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the ‘numerous’ and ‘fewest’ literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a ‘!’. Note that ‘!numerous’ means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., ‘fewest’).

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ‘;’ and are applied in the order specified.

3.6.3.2.2 Default

- `"last"`

3.6.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.6.3.3 ContourOverlap

3.6.3.3.1 Description Controls overlapping contours are treated. The default ‘ignore’ treats overlapping contours as a single contour, regardless of contour orientation. The option ‘honour_opposite_orientations’ makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option ‘overlapping_contours_cancel’ ignores orientation and cancels all contour overlap.

3.6.3.3.2 Default

- `"ignore"`

3.6.3.3.3 Supported Options

- `"ignore"`
- `"honour_opposite_orientations"`
- `"overlapping_contours_cancel"`
- `"honour_opps"`
- `"overlap_cancel"`

3.6.3.4 Inclusivity

3.6.3.4.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.6.3.4.2 Default

- `"center"`

3.6.3.4.3 Supported Options

- `"center"`
- `"centre"`
- `"planar_corner_inclusive"`
- `"planar_inc"`
- `"planar_corner_exclusive"`
- `"planar_exc"`

3.6.3.5 CalibCurveFileName

3.6.3.5.1 Description The file from which a calibration curve should be read from. The format should be line-based with either 2 or 4 numbers per line. For 2 numbers: (current pixel value) (new pixel value) and for 4 numbers: (current pixel value) (uncertainty) (new pixel value) (uncertainty). Uncertainties refer to the prior number and may be uniformly zero if unknown. Lines beginning with '#' are treated as comments and ignored. The curve is linearly interpolated, and must span the full range of pixel values. This is done to avoid extrapolation within the operation since the correct behaviour will differ depending on the specifics of the calibration.

3.6.3.5.2 Default

- `""`

3.6.3.5.3 Examples

- `"/tmp/calib.dat"`

3.6.3.6 NormalizedROILabelRegex

3.6.3.6.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.6.3.6.2 Default

- `".*"`

3.6.3.6.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.6.3.7 ROILabelRegex

3.6.3.7.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single)

regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.6.3.7.2 Default

- `".*"`

3.6.3.7.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.7 AutoCropImages

3.7.1 Description

This operation crops image slices using image-specific metadata embedded within the image.

3.7.2 Parameters

- ImageSelection
- DICOMMargin
- RTIMAGE

3.7.2.1 ImageSelection

3.7.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex

logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.7.2.1.2 Default

- `"all"`

3.7.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.7.2.2 DICOMMargin

3.7.2.2.1 Description The amount of margin (in the DICOM coordinate system) to spare from cropping.

3.7.2.2.2 Default

- `"0.0"`

3.7.2.2.3 Examples

- `"0.1"`
- `"2.0"`
- `"-0.5"`
- `"20.0"`

3.7.2.3 RTIMAGE

3.7.2.3.1 Description If true, attempt to crop the image using information embedded in an RTIMAGE. This option cannot be used with the other options.

3.7.2.3.2 Default

- `"true"`

3.7.2.3.3 Examples

- `"true"`
- `"false"`

3.8 Average

3.8.1 Description

This operation averages image arrays/volumes. It can average over spatial or temporal dimensions. However, rather than relying specifically on time for temporal averaging, any images that have overlapping voxels can be averaged.

3.8.2 Notes

- This operation is typically used to create an aggregate view of a large volume of data. It may also increase SNR and can be used for contouring purposes.

3.8.3 Parameters

- ImageSelection
- AveragingMethod

3.8.3.1 ImageSelection

3.8.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex

logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.8.3.1.2 Default

- `"last"`

3.8.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.8.3.2 AveragingMethod

3.8.3.2.1 Description The averaging method to use. Valid methods are 'overlapping-spatially' and 'overlapping-temporally'.

3.8.3.2.2 Default

- `""`

3.8.3.2.3 Supported Options

- `"overlapping-spatially"`
- `"overlapping-temporally"`

3.9 BCCAExtractRadiomicFeatures

3.9.1 Description

This operation extracts radiomic features from an image and one or more ROIs.

3.9.2 Notes

- This is a 'simplified' version of the full radiomics extract routine that uses defaults that are expected to be reasonable across a wide range of scenarios.

3.9.3 Parameters

- NormalizedROILabelRegex
- ROILabelRegex
- FractionalAreaTolerance
- SimplificationMethod
- UserComment
- FeaturesFileName
- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- ScaleFactor
- ImageFileName
- ColourMapRegex
- WindowLow
- WindowHigh

3.9.3.1 NormalizedROILabelRegex

3.9.3.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.9.3.1.2 Default

- ".*"

3.9.3.1.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.9.3.2 ROILabelRegex

3.9.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.9.3.2.2 Default

- ".*"

3.9.3.2.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.9.3.3 FractionalAreaTolerance

3.9.3.3.1 Description The fraction of area each contour will tolerate during simplified. This is a measure of how much the contour area can change due to simplification.

3.9.3.3.2 Default

- "0.05"

3.9.3.3.3 Examples

- `"0.001"`
- `"0.01"`
- `"0.02"`
- `"0.05"`
- `"0.10"`

3.9.3.4 SimplificationMethod

3.9.3.4.1 Description The specific algorithm used to perform contour simplification. 'Vertex removal' is a simple algorithm that removes vertices one-by-one without replacement. It iteratively ranks vertices and removes the single vertex that has the least impact on contour area. It is best suited to removing redundant vertices or whenever new vertices should not be added. 'Vertex collapse' combines two adjacent vertices into a single vertex at their midpoint. It iteratively ranks vertex pairs and removes the single vertex that has the least total impact on contour area. Note that small sharp features that alternate inward and outward will have a small total area cost, so will be pruned early. Thus this technique acts as a low-pass filter and will defer simplification of high-curvature regions until necessary. It is more economical compared to vertex removal in that it will usually simplify contours more for a given tolerance (or, equivalently, can retain contour fidelity better than vertex removal for the same number of vertices). However, vertex collapse performs an averaging that may result in numerical imprecision.

3.9.3.4.2 Default

- `"vert-rem"`

3.9.3.4.3 Supported Options

- `"vertex-collapse"`
- `"vertex-removal"`

3.9.3.5 UserComment

3.9.3.5.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data.

3.9.3.5.2 Default

- `""`

3.9.3.5.3 Examples

- ""
- "Using XYZ"
- "Patient treatment plan C"

3.9.3.6 FeaturesFileName

3.9.3.6.1 Description Features will be appended to this file. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.9.3.6.2 Default

- ""

3.9.3.6.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.9.3.7 ImageSelection

3.9.3.7.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.9.3.7.2 Default

- `"last"`

3.9.3.7.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.9.3.8 NormalizedROILabelRegex

3.9.3.8.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.9.3.8.2 Default

- `".*"`

3.9.3.8.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`

- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.9.3.9 ROILabelRegex

3.9.3.9.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.9.3.9.2 Default

- ".*"

3.9.3.9.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.9.3.10 ScaleFactor

3.9.3.10.1 Description This factor is applied to the image width and height to magnify (larger than 1) or shrink (less than 1) the image. This factor only affects the output image size. Note that aspect ratio is retained, but rounding for non-integer factors may lead to small (1-2 pixel) discrepancies.

3.9.3.10.2 Default

- "1.5"

3.9.3.10.3 Examples

- "0.5"
- "1.0"

- "2.0"
- "5.23"

3.9.3.11 ImageFileName

3.9.3.11.1 Description The file name to use for the image. If blank, a filename will be generated sequentially.

3.9.3.11.2 Default

- ""

3.9.3.11.3 Examples

- ""
- "/tmp/an_image.png"
- "afile.png"

3.9.3.12 ColourMapRegex

3.9.3.12.1 Description The colour mapping to apply to the image if there is a single channel. The default will match the first available, and if there is no matching map found, the first available will be selected.

3.9.3.12.2 Default

- ".*"

3.9.3.12.3 Supported Options

- "Viridis"
- "Magma"
- "Plasma"
- "Inferno"
- "Jet"
- "MorelandBlueRed"
- "MorelandBlackBody"
- "MorelandExtendedBlackBody"
- "KRC"
- "ExtendedKRC"
- "Kovesi_LinKRYW_5-100_c64"
- "Kovesi_LinKRYW_0-100_c71"
- "Kovesi_Cyclic_cet-c2"
- "LANLOliveGreentoBlue"
- "YgorIncandescent"
- "LinearRamp"

3.9.3.13 WindowLow

3.9.3.13.1 Description If provided, this parameter will override any existing window and level. All pixels with the intensity value or lower will be assigned the lowest possible colour according to the colour map. Not providing a valid number will disable window overrides.

3.9.3.13.2 Default

- `""`

3.9.3.13.3 Examples

- `""`
- `"-1.23"`
- `"0"`
- `"1E4"`

3.9.3.14 WindowHigh

3.9.3.14.1 Description If provided, this parameter will override any existing window and level. All pixels with the intensity value or higher will be assigned the highest possible colour according to the colour map. Not providing a valid number will disable window overrides.

3.9.3.14.2 Default

- `""`

3.9.3.14.3 Examples

- `""`
- `"1.23"`
- `"0"`
- `"10.3E4"`

3.10 BEDConvert

3.10.1 Description

This operation performs Biologically Effective Dose (BED) and Equivalent Dose with 'x'-dose per fraction (EQDx) conversions. Currently, only photon external beam therapy conversions are supported.

3.10.2 Notes

- For an 'EQD2' transformation, select an EQDx conversion model with 2 Gy per fraction (i.e., $x = 2$).
- This operation treats all tissue as either early-responding (e.g., tumour) or late-responding (e.g., some normal tissues). A single alpha/beta estimate for each type (early or late) can be provided. Currently, only two tissue types can be specified.
- This operation requires specification of the initial number of fractions and cannot use dose per fraction. The rationale is that for some models, the dose per fraction would need to be specified for *each individual voxel* since the prescription dose per fraction is **not** the same for voxels outside the PTV.
- Be careful in handling the output of a BED calculation. In particular, BED doses with a given α/β should **only** be summed with BED doses that have the same α/β.

3.10.3 Parameters

- ImageSelection
- AlphaBetaRatioLate
- AlphaBetaRatioEarly
- PriorNumberOfFractions
- PriorPrescriptionDose
- TargetDosePerFraction
- Model
- EarlyROILabelRegex
- EarlyNormalizedROILabelRegex

3.10.3.1 ImageSelection

3.10.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.10.3.1.2 Default

- `"last"`

3.10.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.10.3.2 AlphaBetaRatioLate

3.10.3.2.1 Description The value to use for alpha/beta in late-responding (i.e., 'normal', non-cancerous) tissues. Generally a value of 3.0 Gy is used. Tissues that are sensitive to fractionation may warrant smaller ratios, such as 1.5-3 Gy for cervical central nervous tissues and 2.3-4.9 for lumbar central nervous tissues (consult table 8.1, page 107 in: Joiner et al., 'Fractionation: the linear-quadratic approach', 4th Ed., 2009, in the book 'Basic Clinical Radiobiology', ISBN: 0340929669). Note that the selected ROIs denote early-responding tissues; all remaining tissues are considered late-responding.

3.10.3.2.2 Default

- `"3.0"`

3.10.3.2.3 Examples

- `"2.0"`
- `"3.0"`

3.10.3.3 AlphaBetaRatioEarly

3.10.3.3.1 Description The value to use for alpha/beta in early-responding tissues (i.e., tumourous and some normal tissues). Generally a value of 10.0 Gy is used. Note that the selected ROIs denote early-responding tissues; all remaining tissues are considered late-responding.

3.10.3.3.2 Default

- `"10.0"`

3.10.3.3.3 Examples

- `"10.0"`

3.10.3.4 PriorNumberOfFractions

3.10.3.4.1 Description The number of fractions over which the dose distribution was (or will be) delivered. This parameter is required for both BED and EQDx conversions. Decimal fractions are supported to accommodate multi-pass BED conversions.

3.10.3.4.2 Default

- `"35"`

3.10.3.4.3 Examples

- `"10"`
- `"20.5"`
- `"35"`
- `"40.123"`

3.10.3.5 PriorPrescriptionDose

3.10.3.5.1 Description The prescription dose that was (or will be) delivered to the PTV. This parameter is only used for the 'eqdx-lq-simple-pinned' model. Note that this is a theoretical dose since the PTV or CTV will only nominally receive this dose. Also note that the specified dose need not exist somewhere in the image. It can be purely theoretical to accommodate previous BED conversions.

3.10.3.5.2 Default

- `"70"`

3.10.3.5.3 Examples

- `"15"`
- `"22.5"`
- `"45.0"`
- `"66"`
- `"70.001"`

3.10.3.6 TargetDosePerFraction

3.10.3.6.1 Description The desired dose per fraction 'x' for an EQDx conversion. For an 'EQD2' conversion, this value *must* be 2 Gy. For an 'EQD3.5' conversion, this value should be 3.5 Gy. Note that the specific interpretation of this parameter depends on the model.

3.10.3.6.2 Default

- `"2.0"`

3.10.3.6.3 Examples

- `"1.8"`
- `"2.0"`
- `"5.0"`
- `"8.0"`

3.10.3.7 Model

3.10.3.7.1 Description The BED or EQDx model to use. All assume e was delivered using photon external beam therapy. Current options are 'bed-lq-simple', 'eqdx-lq-simple', and 'eqdx-lq-simple-pinned'. The 'bed-lq-simple' model uses a standard linear-quadratic model that disregards time delays, including repopulation ($BED = (1 + \alpha/\beta)nd$). The 'eqdx-lq-simple' model uses the widely-known, standard formula $EQD_x = nd(d + \alpha/\beta)/(x + \alpha/\beta)$ which is dervied from the linear-quadratic radiobiological model and is also known as the 'Withers' formula. This model disregards time delays, including repopulation. The 'eqdx-lq-simple-pinned' model is an **experimental** alternative to the 'eqdx-lq-simple' model. The 'eqdx-lq-simple-pinned' model implements the 'eqdx-lq-simple' model, but avoids having to specify x dose per fraction. First the prescription dose is transformed to EQDx with x dose per fraction and the effective number of fractions is extracted. Then, each voxel is transformed assuming this effective number of fractions rather than a specific dose per fraction. This model conveniently avoids having to awkwardly specify x dose per fraction for voxels that receive less than x dose. It is also idempotent. Note, however, that the 'eqdx-lq-simple-pinned' model produces EQDx estimates that are **incompatbile** with 'eqdx-lq-simple' EQDx estimates.

3.10.3.7.2 Default

- `"eqdx-lq-simple"`

3.10.3.7.3 Supported Options

- `"bed-lq-simple"`
- `"eqdx-lq-simple"`
- `"eqdx-lq-simple-pinned"`

3.10.3.8 EarlyROILabelRegex

3.10.3.8.1 Description This parameter selects ROI labels/names to consider as bounding early-responding tissues. A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.10.3.8.2 Default

- `".*"`

3.10.3.8.3 Examples

- `".*"`
- `".*GTV.*"`
- `"PTV66"`
- `".*PTV.*|.*GTV.**"`

3.10.3.9 EarlyNormalizedROILabelRegex

3.10.3.9.1 Description This parameter selects ROI labels/names to consider as bounding early-responding tissues. A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.10.3.9.2 Default

- `".*"`

3.10.3.9.3 Examples

- `".*"`
- `".*GTV.*"`
- `"PTV66"`
- `".*PTV.*|.*GTV.**"`

3.11 BoostSerializeDrover

3.11.1 Description

This operation exports all loaded state to a serialized format that can be loaded again later. Is is especially useful for suspending long-running operations with intermittant interactive sub-operations.

3.11.2 Parameters

- Filename
- Components

3.11.2.1 Filename

3.11.2.1.1 Description The filename (or full path name) to which the serialized data should be written. The file format is gzipped XML, which should be portable across most CPUs.

3.11.2.1.2 Default

- `"/tmp/boost_serialized_drover.xml.gz"`

3.11.2.1.3 Examples

- `"/tmp/out.xml.gz"`
- `"./out.xml.gz"`
- `"out.xml.gz"`

3.11.2.2 Components

3.11.2.2.1 Description Which components to include in the output. Currently, any combination of (all images), (all contours), (all point clouds), (all surface meshes), and (all treatment plans) can be selected. Note that RTDOSEs are treated as images.

3.11.2.2.2 Default

- `"images+contours+pointclouds+surfacemeshes+tplans"`

3.11.2.2.3 Examples

- `"images"`
- `"images+pointclouds"`
- `"images+pointclouds+surfacemeshes"`
- `"pointclouds+surfacemeshes"`
- `"tplans+images+contours"`
- `"contours+images+pointclouds"`

3.12 BuildLexiconInteractively

3.12.1 Description

This operation interactively builds a lexicon using the currently loaded contour labels. It is useful for constructing a domain-specific lexicon from a set of representative data.

3.12.2 Notes

- An exclusive approach is taken for ROI selection rather than an inclusive approach because regex negations are not easily supported in the POSIX syntax.

3.12.3 Parameters

- CleanLabels
- JunkLabel
- OmitROILabelRegex
- LexiconSeedFile

3.12.3.1 CleanLabels

3.12.3.1.1 Description A listing of the labels of interest. These will be (some of) the 'clean' entries in the finished lexicon. You should only name ROIs you specifically care about and which have a single, unambiguous occurence in the data set (e.g., 'Left_Parotid' is good, but 'JUNK' and 'Parotids' are bad

– you won't be able to select the single 'JUNK' label if all you care about are parotids.

3.12.3.1.2 Default

- `"Body,Brainstem,Chiasm,Cord,Larynx Pharynx,Left Eye,Left Optic Nerve,Left Parotid,Left Submand,Left Temp Lobe,Oral Cavity,Right Eye,Right Optic Nerve,Right Parotid,Right Submand,Right Temp Lobe"`

3.12.3.1.3 Examples

- `"Left Parotid,Right Parotid,Left Submand,Right Submand"`
- `"Left Submand,Right Submand"`

3.12.3.2 JunkLabel

3.12.3.2.1 Description A label to apply to the un-matched labels. This helps prevent false positives by excluding names which are close to a desired clean label. For example, if you are looking for 'Left_Parotid' you will want to mark 'left-parotid_opti' and 'OLDLeftParotid' as junk. Passing an empty string disables junk labeling.

3.12.3.2.2 Default

- `"JUNK"`

3.12.3.2.3 Examples

- `""`
- `"Junk"`
- `"Irrelevant"`
- `"NA_Organ"`

3.12.3.3 OmitROILabelRegex

3.12.3.3.1 Description This parameter selects ROI labels/names to prune. Only matching ROIs will be pruned. The default will match no ROIs. A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.12.3.3.2 Default

- ""

3.12.3.3.3 Examples

- ".*left.*|.*right.*|.*eyes.*"
- ".*PTV.*|.*CTV.*|.*GTV.*"

3.12.3.4 LexiconSeedFile

3.12.3.4.1 Description A file containing a 'seed' lexicon to use and add to. This is the lexicon that is being built. It will be modified.

3.12.3.4.2 Default

- ""

3.12.3.4.3 Examples

- "./some_lexicon"
- "/tmp/temp_lexicon"

3.13 CT_Liver_Perfusion

3.13.1 Description

This operation performed dynamic contrast-enhanced CT perfusion image modeling on a time series image volume.

3.13.2 Notes

- This routine is used for research purposes only.

3.13.3 Parameters

No registered options.

3.14 CT_Liver_Perfusion_First_Run

3.14.1 Description

This operation performed dynamic contrast-enhanced CT perfusion image modeling on a time series image volume.

3.14.2 Notes

- Use this mode when peeking at the data for the first time. It avoids computing much, just lets you *look* at the data, find t_0, etc..

3.14.3 Parameters

No registered options.

3.15 CT_Liver_Perfusion_Ortho_Views

3.15.1 Description

This operation performed dynamic contrast-enhanced CT perfusion image modeling on a time series image volume.

3.15.2 Notes

- Use this mode when you are only interested in oblique/orthogonal views. The point of this operation is to keep memory low so image sets can be compared.

3.15.3 Parameters

No registered options.

3.16 CT_Liver_Perfusion_Pharmaco_1C2I_5Param

3.16.1 Description

This operation performed dynamic contrast-enhanced CT perfusion image modeling on a time series image volume.

3.16.2 Parameters

- AIFROINameRegex
- ExponentialKernelCoeffTruncation
- FastChebyshevMultiplication
- PlotAIFVIF
- PlotPixelModel
- PreDecimateOutSizeR
- PreDecimateOutSizeC
- TargetROINameRegex
- UseBasisSplineInterpolation
- BasisSplineCoefficients

- BasisSplineOrder
- UseChebyshevPolyMethod
- ChebyshevPolyCoefficients
- VIFROINameRegex

3.16.2.1 AIFROINameRegex

3.16.2.1.1 Description Regex for the name of the ROI to use as the AIF. It should generally be a major artery near the trunk or near the tissue of interest.

3.16.2.1.2 Default

- `"Abdominal_Aorta"`

3.16.2.1.3 Examples

- `"Abdominal_Aorta"`
- `".*Aorta.*"`
- `"Major_Artery"`

3.16.2.2 ExponentialKernelCoeffTruncation

3.16.2.2.1 Description Control the number of Chebyshev coefficients used to approximate the exponential kernel. Usually ~10 will suffice. ~20 is probably overkill, and ~5 is probably too few. It is probably better to err on the side of caution and enlarge this number if you're worried about loss of precision – this will slow the computation somewhat. (You might be able to offset by retaining fewer coefficients in Chebyshev multiplication; see 'FastChebyshevMultiplication' parameter.)

3.16.2.2.2 Default

- `"10"`

3.16.2.2.3 Examples

- `"20"`
- `"15"`
- `"10"`
- `"5"`

3.16.2.3 FastChebyshevMultiplication

3.16.2.3.1 Description Control coefficient truncation/pruning to speed up Chebyshev polynomial multiplication. (This setting does nothing if the Chebyshev method is not being used.) The choice of this number depends on how much precision you are willing to forgo. It also strongly depends on the number of datum in the AIF, VIF, and the number of coefficients used to approximate the exponential kernel (usually ~10 suffices). Numbers are specified relative to max(N,M), where N and M are the number of coefficients in the two Chebyshev expansions taking part in the multiplication. If too many coefficients are requested (i.e., more than (N+M-2)) then the full non-approximate multiplication is carried out.

3.16.2.3.2 Default

- `"*10000000.0"`

3.16.2.3.3 Examples

- `"*2.0"`
- `"*1.5"`
- `"*1.0"`
- `"*0.5"`
- `"*0.3"`

3.16.2.4 PlotAIFVIF

3.16.2.4.1 Description Control whether the AIF and VIF should be shown prior to modeling.

3.16.2.4.2 Default

- `"false"`

3.16.2.4.3 Examples

- `"true"`
- `"false"`

3.16.2.5 PlotPixelModel

3.16.2.5.1 Description Show a plot of the fitted model for a specified pixel. Plotting happens immediately after the pixel is processed. You can supply arbitrary metadata, but must also supply Row and Column numbers. Note that numerical comparisons are performed lexically, so you have to be exact. Also note the sub-separation token is a semi-colon, not a colon.

3.16.2.5.2 Default

- ""

3.16.2.5.3 Examples

- "Row@12;Column@4;Description@.*k1A.*"
- "Row@256;Column@500;SliceLocation@23;SliceThickness@0.5"
- "Row@256;Column@500;Some@thing#Row@256;Column@501;Another@thing"
- "Row@0;Column@5#Row@4;Column@5#Row@8;Column@5#Row@12;Column@5"

3.16.2.6 PreDecimateOutSizeR

3.16.2.6.1 Description The number of pixels along the row unit vector to group into an outgoing pixel. This optional step can reduce computation effort by downsampling (decimating) images before computing fitted parameter maps (but *after* computing AIF and VIF time courses). Must be a multiplicative factor of the incoming image's row count. No decimation occurs if either this or 'PreDecimateOutSizeC' is zero or negative.

3.16.2.6.2 Default

- "8"

3.16.2.6.3 Examples

- "0"
- "2"
- "4"
- "8"
- "16"
- "32"
- "64"
- "128"
- "256"
- "512"

3.16.2.7 PreDecimateOutSizeC

3.16.2.7.1 Description The number of pixels along the column unit vector to group into an outgoing pixel. This optional step can reduce computation effort by downsampling (decimating) images before computing fitted parameter maps (but *after* computing AIF and VIF time courses). Must be a multiplicative factor of the incoming image's column count. No decimation occurs if either this or 'PreDecimateOutSizeR' is zero or negative.

3.16.2.7.2 Default

- "8"

3.16.2.7.3 Examples

- "0"
- "2"
- "4"
- "8"
- "16"
- "32"
- "64"
- "128"
- "256"
- "512"

3.16.2.8 TargetROINameRegex

3.16.2.8.1 Description Regex for the name of the ROI to perform modeling within. The largest contour is usually what you want, but you can also be more focused.

3.16.2.8.2 Default

- ".*Body.*"

3.16.2.8.3 Examples

- "Liver_Patches_For_Testing_Smaller"
- "Liver_Patches_For_Testing"
- "Suspected_Liver_Rough"
- "Rough_Body"
- ".*body.*"
- ".*something.*\|.*another.*thing.*"

3.16.2.9 UseBasisSplineInterpolation

3.16.2.9.1 Description Control whether the AIF and VIF should use basis spline interpolation in conjunction with the Chebyshev polynomial method. If this option is not set, linear interpolation is used instead. Linear interpolation may result in a less-smooth AIF and VIF (and therefore possibly slower optimizer convergence), but is safer if you cannot verify the AIF and VIF plots are reasonable. This option currently produces an effect only if the Chebyshev polynomial method is being used.

3.16.2.9.2 Default

- `"false"`

3.16.2.9.3 Examples

- `"true"`
- `"false"`

3.16.2.10 BasisSplineCoefficients

3.16.2.10.1 Description Control the number of basis spline coefficients to use, if applicable. (This setting does nothing when basis splines are not being used.) Valid options for this setting depend on the amount of data and b-spline order. This number controls the number of coefficients that are fitted (via least-squares). You must verify that overfitting is not happening. If in doubt, use fewer coefficients. There are two ways to specify the number: relative and absolute. Relative means relative to the number of datum. For example, if the AIF and VIF have ~40 datum then generally '*0.5' is safe. (*'0.5' means there are half the number of coefficients as datum.) Inspect for overfitting and poor fit. Because this routine happens once and is fast, do not tweak to optimize for speed; the aim of this method is to produce a smooth and accurate AIF and VIF. Because an integer number of coefficients are needed, so rounding is used. You can also specify the absolute number of coefficients to use like '20'. It often makes more sense to use relative specification. Be aware that not all inputs can be honoured due to limits on b-spline knots and breaks, and may cause unpredictable behaviour or internal failure.

3.16.2.10.2 Default

- `"*0.5"`

3.16.2.10.3 Examples

- `"*0.8"`
- `"*0.5"`
- `"*0.3"`
- `"20.0"`
- `"10.0"`

3.16.2.11 BasisSplineOrder

3.16.2.11.1 Description Control the polynomial order of basis spline interpolation to use, if applicable. (This setting does nothing when basis splines are not being used.) This parameter controls the order of polynomial used for b-spline interpolation, and therefore has ramifications for the computability and

numerical stability of AIF and VIF derivatives. Stick with '4' or '5' if you're unsure.

3.16.2.11.2 Default

- `"4"`

3.16.2.11.3 Examples

- `"1"`
- `"2"`
- `"3"`
- `"4"`
- `"5"`
- `"6"`
- `"7"`
- `"8"`
- `"9"`
- `"10"`

3.16.2.12 UseChebyshevPolyMethod

3.16.2.12.1 Description Control whether the AIF and VIF should be approximated by Chebyshev polynomials. If this option is not set, a inear interpolation approach is used instead.

3.16.2.12.2 Default

- `"true"`

3.16.2.12.3 Examples

- `"true"`
- `"false"`

3.16.2.13 ChebyshevPolyCoefficients

3.16.2.13.1 Description Control the number of Chebyshev polynomial coefficients to use, if applicable. (This setting does nothing when the Chebyshev polynomial method is not being used.) This number controls the number of coefficients that are computed. There are two ways to specify the number: relative and absolute. Relative means relative to the number of datum. For example, if the AIF and VIF have ~40 datum then generally '*2' is safe. (*'2' means there are 2x the number of coefficients as datum; usually overkill.) A good middle-ground is '*1' which is faster but should produce similar results. For speed '/2' is even faster, but can produce bad results in some cases. Because

an integer number of coefficients are needed, rounding is used. You can also specify the absolute number of coefficients to use like '20'. It often makes more sense to use relative specification. Be aware that not all inputs can be honoured (i.e., too large, too small, or negative), and may cause unpredictable behaviour or internal failure.

3.16.2.13.2 Default

- `"*2.0"`

3.16.2.13.3 Examples

- `"*10.0"`
- `"*5.0"`
- `"*2.0"`
- `"*1.23"`
- `"*1.0"`
- `"/1.0"`
- `"/2.0"`
- `"/3.0"`
- `"/5.0"`
- `"100.0"`
- `"50.0"`
- `"20"`
- `"10.01"`

3.16.2.14 VIFROINameRegex

3.16.2.14.1 Description Regex for the name of the ROI to use as the VIF. It should generally be a major vein near the trunk or near the tissue of interest.

3.16.2.14.2 Default

- `"Hepatic_Portal_Vein"`

3.16.2.14.3 Examples

- `"Hepatic_Portal_Vein"`
- `".*Portal.*Vein.*"`
- `"Major_Vein"`

3.17 CT_Liver_Perfusion_Pharmaco_1C2I_Reduced3Param

3.17.1 Description

This operation performed dynamic contrast-enhanced CT perfusion image modeling on a time series image volume.

3.17.2 Parameters

- AIFROINameRegex
- ExponentialKernelCoeffTruncation
- FastChebyshevMultiplication
- PlotAIFVIF
- PlotPixelModel
- PreDecimateOutSizeR
- PreDecimateOutSizeC
- TargetROINameRegex
- UseBasisSplineInterpolation
- BasisSplineCoefficients
- BasisSplineOrder
- ChebyshevPolyCoefficients
- VIFROINameRegex

3.17.2.1 AIFROINameRegex

3.17.2.1.1 Description Regex for the name of the ROI to use as the AIF. It should generally be a major artery near the trunk or near the tissue of interest.

3.17.2.1.2 Default

- `"Abdominal_Aorta"`

3.17.2.1.3 Examples

- `"Abdominal_Aorta"`
- `".*Aorta.*"`
- `"Major_Artery"`

3.17.2.2 ExponentialKernelCoeffTruncation

3.17.2.2.1 Description Control the number of Chebyshev coefficients used to approximate the exponential kernel. Usually ~10 will suffice. ~20 is probably overkill, and ~5 is probably too few. It is probably better to err on the side of caution and enlarge this number if you're worried about loss of precision – this will slow the computation somewhat. (You might be able to offset by retaining fewer coefficients in Chebyshev multiplication; see 'FastChebyshevMultiplication' parameter.)

3.17.2.2.2 Default

- `"10"`

3.17.2.2.3 Examples

- `"20"`
- `"15"`
- `"10"`
- `"5"`

3.17.2.3 FastChebyshevMultiplication

3.17.2.3.1 Description Control coefficient truncation/pruning to speed up Chebyshev polynomial multiplication. (This setting does nothing if the Chebyshev method is not being used.) The choice of this number depends on how much precision you are willing to forgo. It also strongly depends on the number of datum in the AIF, VIF, and the number of coefficients used to approximate the exponential kernel (usually ~10 suffices). Numbers are specified relative to max(N,M), where N and M are the number of coefficients in the two Chebyshev expansions taking part in the multiplication. If too many coefficients are requested (i.e., more than (N+M-2)) then the full non-approximate multiplication is carried out.

3.17.2.3.2 Default

- `"*10000000.0"`

3.17.2.3.3 Examples

- `"*2.0"`
- `"*1.5"`
- `"*1.0"`
- `"*0.5"`
- `"*0.3"`

3.17.2.4 PlotAIFVIF

3.17.2.4.1 Description Control whether the AIF and VIF should be shown prior to modeling.

3.17.2.4.2 Default

- `"false"`

3.17.2.4.3 Examples

- `"true"`
- `"false"`

3.17.2.5 PlotPixelModel

3.17.2.5.1 Description Show a plot of the fitted model for a specified pixel. Plotting happens immediately after the pixel is processed. You can supply arbitrary metadata, but must also supply Row and Column numbers. Note that numerical comparisons are performed lexically, so you have to be exact. Also note the sub-separation token is a semi-colon, not a colon.

3.17.2.5.2 Default

- `""`

3.17.2.5.3 Examples

- `"Row@12;Column@4;Description@.*k1A.*"`
- `"Row@256;Column@500;SliceLocation@23;SliceThickness@0.5"`
- `"Row@256;Column@500;Some@thing#Row@256;Column@501;Another@thing"`
- `"Row@0;Column@5#Row@4;Column@5#Row@8;Column@5#Row@12;Column@5"`

3.17.2.6 PreDecimateOutSizeR

3.17.2.6.1 Description The number of pixels along the row unit vector to group into an outgoing pixel. This optional step can reduce computation effort by downsampling (decimating) images before computing fitted parameter maps (but *after* computing AIF and VIF time courses). Must be a multiplicative factor of the incoming image's row count. No decimation occurs if either this or 'PreDecimateOutSizeC' is zero or negative.

3.17.2.6.2 Default

- `"8"`

3.17.2.6.3 Examples

- `"0"`
- `"2"`
- `"4"`
- `"8"`
- `"16"`
- `"32"`
- `"64"`
- `"128"`

- "256"
- "512"

3.17.2.7 PreDecimateOutSizeC

3.17.2.7.1 Description The number of pixels along the column unit vector to group into an outgoing pixel. This optional step can reduce computation effort by downsampling (decimating) images before computing fitted parameter maps (but *after* computing AIF and VIF time courses). Must be a multiplicative factor of the incoming image's column count. No decimation occurs if either this or 'PreDecimateOutSizeR' is zero or negative.

3.17.2.7.2 Default

- "8"

3.17.2.7.3 Examples

- "0"
- "2"
- "4"
- "8"
- "16"
- "32"
- "64"
- "128"
- "256"
- "512"

3.17.2.8 TargetROINameRegex

3.17.2.8.1 Description Regex for the name of the ROI to perform modeling within. The largest contour is usually what you want, but you can also be more focused.

3.17.2.8.2 Default

- ".*Body.*"

3.17.2.8.3 Examples

- "Liver_Patches_For_Testing_Smaller"
- "Liver_Patches_For_Testing"
- "Suspected_Liver_Rough"
- "Rough_Body"
- ".*body.*"
- ".*something.*\|.*another.*thing.*"

3.17.2.9 UseBasisSplineInterpolation

3.17.2.9.1 Description Control whether the AIF and VIF should use basis spline interpolation in conjunction with the Chebyshev polynomial method. If this option is not set, linear interpolation is used instead. Linear interpolation may result in a less-smooth AIF and VIF (and therefore possibly slower optimizer convergence), but is safer if you cannot verify the AIF and VIF plots are reasonable. This option currently produces an effect only if the Chebyshev polynomial method is being used.

3.17.2.9.2 Default

- `"false"`

3.17.2.9.3 Examples

- `"true"`
- `"false"`

3.17.2.10 BasisSplineCoefficients

3.17.2.10.1 Description Control the number of basis spline coefficients to use, if applicable. (This setting does nothing when basis splines are not being used.) Valid options for this setting depend on the amount of data and b-spline order. This number controls the number of coefficients that are fitted (via least-squares). You must verify that overfitting is not happening. If in doubt, use fewer coefficients. There are two ways to specify the number: relative and absolute. Relative means relative to the number of datum. For example, if the AIF and VIF have ~40 datum then generally '*0.5' is safe. (*'0.5' means there are half the number of coefficients as datum.) Inspect for overfitting and poor fit. Because this routine happens once and is fast, do not tweak to optimize for speed; the aim of this method is to produce a smooth and accurate AIF and VIF. Because an integer number of coefficients are needed, so rounding is used. You can also specify the absolute number of coefficients to use like '20'. It often makes more sense to use relative specification. Be aware that not all inputs can be honoured due to limits on b-spline knots and breaks, and may cause unpredictable behaviour or internal failure.

3.17.2.10.2 Default

- `"*0.5"`

3.17.2.10.3 Examples

- `"*0.8"`
- `"*0.5"`

- "*0.3"
- "20.0"
- "10.0"

3.17.2.11 BasisSplineOrder

3.17.2.11.1 Description Control the polynomial order of basis spline interpolation to use, if applicable. (This setting does nothing when basis splines are not being used.) This parameter controls the order of polynomial used for b-spline interpolation, and therefore has ramifications for the computability and numerical stability of AIF and VIF derivatives. Stick with '4' or '5' if you're unsure.

3.17.2.11.2 Default

- "4"

3.17.2.11.3 Examples

- "1"
- "2"
- "3"
- "4"
- "5"
- "6"
- "7"
- "8"
- "9"
- "10"

3.17.2.12 ChebyshevPolyCoefficients

3.17.2.12.1 Description Control the number of Chebyshev polynomial coefficients to use, if applicable. (This setting does nothing when the Chebyshev polynomial method is not being used.) This number controls the number of coefficients that are computed. There are two ways to specify the number: relative and absolute. Relative means relative to the number of datum. For example, if the AIF and VIF have ~40 datum then generally '*2' is safe. ('*2' means there are 2x the number of coefficients as datum; usually overkill.) A good middle-ground is '*1' which is faster but should produce similar results. For speed '/2' is even faster, but can produce bad results in some cases. Because an integer number of coefficients are needed, rounding is used. You can also specify the absolute number of coefficients to use like '20'. It often makes more sense to use relative specification. Be aware that not all inputs can be honoured (i.e., too large, too small, or negative), and may cause unpredictable behaviour or internal failure.

3.17.2.12.2 Default

- `"*2.0"`

3.17.2.12.3 Examples

- `"*10.0"`
- `"*5.0"`
- `"*2.0"`
- `"*1.23"`
- `"*1.0"`
- `"/1.0"`
- `"/2.0"`
- `"/3.0"`
- `"/5.0"`
- `"100.0"`
- `"50.0"`
- `"20"`
- `"10.01"`

3.17.2.13 VIFROINameRegex

3.17.2.13.1 Description Regex for the name of the ROI to use as the VIF. It should generally be a major vein near the trunk or near the tissue of interest.

3.17.2.13.2 Default

- `"Hepatic_Portal_Vein"`

3.17.2.13.3 Examples

- `"Hepatic_Portal_Vein"`
- `".*Portal.*Vein.*"`
- `"Major_Vein"`

3.18 CellularAutomata

3.18.1 Description

This operation implements 2D cellular automata (Conway's Game of Life) with periodic boundary conditions.

3.18.2 Notes

- The provided image collection must be rectilinear. All images will be modeled independently of one another.

3.18.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Channel
- Method
- Iterations
- Low
- High

3.18.3.1 ImageSelection

3.18.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.18.3.1.2 Default

- `"last"`

3.18.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`

- "none"
- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.18.3.2 NormalizedROILabelRegex

3.18.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.18.3.2.2 Default

- ".*"

3.18.3.2.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.18.3.3 ROILabelRegex

3.18.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping

to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.18.3.3.2 Default

- `".*"`

3.18.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.18.3.4 Channel

3.18.3.4.1 Description The channel to operated on (zero-based). Negative values will cause all channels to be operated on.

3.18.3.4.2 Default

- `"0"`

3.18.3.4.3 Examples

- `"-1"`
- `"0"`
- `"1"`

3.18.3.5 Method

3.18.3.5.1 Description Controls the type of automata to simulate.

3.18.3.5.2 Default

- `"conway's-game-of-life"`

3.18.3.5.3 Supported Options

- `"conway's-game-of-life"`
- `"gravity-down"`
- `"gravity-up"`
- `"gravity-left"`
- `"gravity-right"`
- `"gravity-in"`
- `"gravity-out"`

3.18.3.6 Iterations

3.18.3.6.1 Description The number of iterations to simulate. Note that intermediary iterations are not retained.

3.18.3.6.2 Default

- `"1"`

3.18.3.6.3 Examples

- `"1"`
- `"10"`
- `"1000"`

3.18.3.7 Low

3.18.3.7.1 Description The voxel value that represents 'dead' cells. Since cells are either 'alive' or 'dead', the value halfway between the low and high values is used as the threshold.

3.18.3.7.2 Default

- `"0.0"`

3.18.3.7.3 Examples

- `"0.0"`
- `"-1.23"`
- `"10.0"`

3.18.3.8 High

3.18.3.8.1 Description The voxel value that represents 'alive' cells. Since cells are either 'alive' or 'dead', the value halfway between the low and high values is used as the threshold.

3.18.3.8.2 Default

- `"1.0"`

3.18.3.8.3 Examples

- `"1.5"`
- `"-0.23"`
- `"255.0"`

3.19 ClusterDBSCAN

3.19.1 Description

This routine performs DBSCAN clustering on an image volume. The clustering is limited within ROI(s) and also within a range of voxel intensities. Voxels values are overwritten with the cluster ID (if applicable) or a generic configurable background value.

3.19.2 Notes

- This operation will work with single images and image volumes. Images need not be rectilinear.

3.19.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- ContourOverlap
- Inclusivity
- Channel
- Lower
- Upper
- MinPoints
- MaxPoints
- Eps
- BackgroundValue
- Reduction

3.19.3.1 ImageSelection

3.19.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is

possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.19.3.1.2 Default

- `"last"`

3.19.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.19.3.2 NormalizedROILabelRegex

3.19.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI

name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.19.3.2.2 Default

- `".*"`

3.19.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.19.3.3 ROILabelRegex

3.19.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.19.3.3.2 Default

- `".*"`

3.19.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`

- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.19.3.4 ContourOverlap

3.19.3.4.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.19.3.4.2 Default

- "ignore"

3.19.3.4.3 Supported Options

- "ignore"
- "honour_opposite_orientations"
- "overlapping_contours_cancel"
- "honour_opps"
- "overlap_cancel"

3.19.3.5 Inclusivity

3.19.3.5.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.19.3.5.2 Default

- "center"

3.19.3.5.3 Supported Options

- "center"
- "centre"
- "planar_corner_inclusive"
- "planar_inc"
- "planar_corner_exclusive"
- "planar_exc"

3.19.3.6 Channel

3.19.3.6.1 Description The channel to operated on (zero-based). Negative values will cause all channels to be operated on.

3.19.3.6.2 Default

- `"0"`

3.19.3.6.3 Examples

- `"-1"`
- `"0"`
- `"1"`

3.19.3.7 Lower

3.19.3.7.1 Description Lower threshold (inclusive) below which voxels will be ignored by this routine.

3.19.3.7.2 Default

- `"-inf"`

3.19.3.7.3 Examples

- `"-inf"`
- `"0.0"`
- `"1024"`

3.19.3.8 Upper

3.19.3.8.1 Description Upper threshold (inclusive) above which voxels will be ignored by this routine.

3.19.3.8.2 Default

- `"inf"`

3.19.3.8.3 Examples

- `"inf"`
- `"1.0"`
- `"2048"`

3.19.3.9 MinPoints

3.19.3.9.1 Description DBSCAN algorithm parameter representing the minimum number of points that must appear in the vicinity for a cluster to be recognized. Sanders, et al. (1998) recommend a default of twice the dimensionality, but what is considered to be a reasonable value depends on the sparsity of the inputs and geometry. For regular grids, a slightly smaller value might be more appropriate.

3.19.3.9.2 Default

- `"5"`

3.19.3.9.3 Examples

- `"4"`
- `"6"`
- `"15"`

3.19.3.10 MaxPoints

3.19.3.10.1 Description Reject clusters if they would contain more than this many members. This parameter can be used to reject irrelevant background clusters or to help search for disconnected clusters. Setting this parameter appropriately will improve both memory usage and runtime considerably.

3.19.3.10.2 Default

- `"inf"`

3.19.3.10.3 Examples

- `"10"`
- `"1000"`
- `"1E6"`
- `"inf"`

3.19.3.11 Eps

3.19.3.11.1 Description DBSCAN algorithm parameter representing the threshold separation distance (in DICOM units; mm) between members of a cluster. All members in a cluster must be separated from at least MinPoints points within a distance of Eps. There is a standard way to determine an optimal value from the data itself, but requires generating a k-nearest-neighbours clustering first, and then visually identifying an appropriate 'kink' in the k-distances plot. This approach is not implemented here. Alternatively, the sparsity of the data and the specific problem domain must be used to estimate a desirable separation Eps.

3.19.3.11.2 Default

- `"4.0"`

3.19.3.11.3 Examples

- `"1.5"`
- `"2.5"`
- `"4.0"`
- `"10.0"`

3.19.3.12 BackgroundValue

3.19.3.12.1 Description The voxel intensity that will be assigned to all voxels that are not members of a cluster. Note that this value can be anything, but cluster numbers are zero-based, so a negative background is probably desired.

3.19.3.12.2 Default

- `"-1.0"`

3.19.3.12.3 Examples

- `"-1.0"`
- `"0.0"`
- `"100.23"`
- `"nan"`
- `"-inf"`

3.19.3.13 Reduction

3.19.3.13.1 Description Voxels within a cluster can be marked as-is, or reduced in a variety of ways. If reduction is not used, voxels in a valid cluster will have their values replaced with the cluster ID number. If 'median' reduction is specified, the component-wise median is reported for each cluster; the x-, y-, and z-coordinates of all voxels in each individual cluster will be reduced to the median coordinate.

3.19.3.13.2 Default

- `"none"`

3.19.3.13.3 Supported Options

- `"none"`
- `"median"`

3.20 ComparePixels

3.20.1 Description

This operation compares images ('test' images and 'reference' images) on a per-voxel/per-pixel basis. Any combination of 2D and 3D images is supported, including images which do not fully overlap, but the reference image array must be rectilinear (this property is verified).

3.20.2 Notes

- Images are overwritten, but ReferenceImages are not. Multiple Images may be specified, but only one ReferenceImages may be specified.
- The reference image array must be rectilinear. (This is a requirement specific to this implementation, a less restrictive implementation could overcome the issue.)
- For the fastest and most accurate results, test and reference image arrays should spatially align. However, alignment is **not** necessary. If test and reference image arrays are aligned, image adjacency can be precomputed and the analysis will be faster. If not, image adjacency must be evaluated for every voxel.
- The distance-to-agreement comparison will tend to overestimate the distance, especially when the DTA value is low, because voxel size effects will dominate the estimation. Reference images should be supersampled as necessary.
- This operation optionally makes use of interpolation for sub-voxel distance estimation. However, interpolation is currently limited to be along the edges connecting nearest- and next-nearest voxel centres. In other words, true volumetric interpolation is **not** available. Implicit interpolation is also used (via the intermediate value theorem) for the distance-to-agreement comparison, which results in distance estimation that may vary up to the largest caliper distance of a voxel. For this reason, the accuracy of all comparisons should be expected to be limited by image spatial resolution (i.e., voxel dimensions). Reference images should be supersampled as necessary.

3.20.3 Parameters

- ImageSelection
- ReferenceImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Method
- Channel
- TestImgLowerThreshold

- TestImgUpperThreshold
- RefImgLowerThreshold
- RefImgUpperThreshold
- DiscType
- DTAVoxValEqAbs
- DTAVoxValEqRelDiff
- DTAMax
- DTAInterpolationMethod
- GammaDTAThreshold
- GammaDiscThreshold
- GammaTerminateAboveOne

3.20.3.1 ImageSelection

3.20.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.20.3.1.2 Default

- `"all"`

3.20.3.1.3 Examples

- `"last"`
- `"first"`

- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.20.3.2 ReferenceImageSelection

3.20.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.20.3.2.2 Default

- `"all"`

3.20.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`

- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.20.3.3 NormalizedROILabelRegex

3.20.3.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.20.3.3.2 Default

- ".*"

3.20.3.3.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.20.3.4 ROILabelRegex

3.20.3.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single)

regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.20.3.4.2 Default

- `".*"`

3.20.3.4.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.20.3.5 Method

3.20.3.5.1 Description The comparison method to compute. Three options are currently available: distance-to-agreement (DTA), discrepancy, and gamma-index. All three are fully 3D, but can also work for 2D or mixed 2D-3D comparisons. DTA is a measure of how far away the nearest voxel (in the reference images) is with a voxel intensity sufficiently close to each voxel in the test images. This comparison ignores pixel intensities except to test if the values match within the specified tolerance. The voxel neighbourhood is exhaustively explored until a suitable voxel is found. Implicit interpolation is used to detect when the value could be found via interpolation, but explicit interpolation is not used. Thus distance might be overestimated. A discrepancy comparison measures the point intensity discrepancy without accounting for spatial shifts. A gamma analysis combines distance-to-agreement and point differences into a single index which is best used to test if both DTA and discrepancy criteria are satisfied (gamma <= 1 iff both pass). It was proposed by Low et al. in 1998 ((doi:10.1118/1.598248). Gamma analyses permits trade-offs between spatial and dosimetric discrepancies which can arise when the image arrays slightly differ in alignment or pixel values.

3.20.3.5.2 Default

- `"gamma-index"`

3.20.3.5.3 Supported Options

- `"gamma-index"`
- `"DTA"`

- "discrepancy"

3.20.3.6 Channel

3.20.3.6.1 Description The channel to compare (zero-based). Note that both test images and reference images will share this specifier.

3.20.3.6.2 Default

- "0"

3.20.3.6.3 Examples

- "0"
- "1"
- "2"

3.20.3.7 TestImgLowerThreshold

3.20.3.7.1 Description Pixel lower threshold for the test images. Only voxels with values above this threshold (inclusive) will be altered.

3.20.3.7.2 Default

- "-inf"

3.20.3.7.3 Examples

- "-inf"
- "0.0"
- "200"

3.20.3.8 TestImgUpperThreshold

3.20.3.8.1 Description Pixel upper threshold for the test images. Only voxels with values below this threshold (inclusive) will be altered.

3.20.3.8.2 Default

- "inf"

3.20.3.8.3 Examples

- "inf"
- "1.23"
- "1000"

3.20.3.9 RefImgLowerThreshold

3.20.3.9.1 Description Pixel lower threshold for the reference images. Only voxels with values above this threshold (inclusive) will be altered.

3.20.3.9.2 Default

- `"-inf"`

3.20.3.9.3 Examples

- `"-inf"`
- `"0.0"`
- `"200"`

3.20.3.10 RefImgUpperThreshold

3.20.3.10.1 Description Pixel upper threshold for the reference images. Only voxels with values below this threshold (inclusive) will be altered.

3.20.3.10.2 Default

- `"inf"`

3.20.3.10.3 Examples

- `"inf"`
- `"1.23"`
- `"1000"`

3.20.3.11 DiscType

3.20.3.11.1 Description Parameter for all comparisons estimating the direct, voxel-to-voxel discrepancy. There are currently three types available. 'Relative' is the absolute value of the difference of two voxel values divided by the largest of the two values. 'Difference' is the difference of two voxel values. 'PinnedToMax' is the absolute value of the difference of two voxel values divided by the largest voxel value in the selected images.

3.20.3.11.2 Default

- `"relative"`

3.20.3.11.3 Supported Options

- `"relative"`
- `"difference"`
- `"pinned-to-max"`

3.20.3.12 DTAVoxValEqAbs

3.20.3.12.1 Description Parameter for all comparisons involving a distance-to-agreement (DTA) search. The difference in voxel values considered to be sufficiently equal (absolute; in voxel intensity units). Note: This value CAN be zero. It is meant to help overcome noise. Note that this value is ignored by all interpolation methods.

3.20.3.12.2 Default

- `"1.0E-3"`

3.20.3.12.3 Examples

- `"1.0E-3"`
- `"1.0E-5"`
- `"0.0"`
- `"0.5"`

3.20.3.13 DTAVoxValEqRelDiff

3.20.3.13.1 Description Parameter for all comparisons involving a distance-to-agreement (DTA) search. The difference in voxel values considered to be sufficiently equal (~relative difference; in %). Note: This value CAN be zero. It is meant to help overcome noise. Note that this value is ignored by all interpolation methods.

3.20.3.13.2 Default

- `"1.0"`

3.20.3.13.3 Examples

- `"0.1"`
- `"1.0"`
- `"10.0"`

3.20.3.14 DTAMax

3.20.3.14.1 Description Parameter for all comparisons involving a distance-to-agreement (DTA) search. Maximally acceptable distance-to-agreement (in DICOM units: mm) above which to stop searching. All voxels within this distance will be searched unless a matching voxel is found. Note that a gamma-index comparison may terminate this search early if the gamma-index is known to be greater than one. It is recommended to make this value approximately 1 voxel width larger than necessary in case a matching voxel can be located near the boundary. Also note that some voxels beyond the DTA_max distance may be evaluated.

3.20.3.14.2 Default

- `"30.0"`

3.20.3.14.3 Examples

- `"3.0"`
- `"5.0"`
- `"50.0"`

3.20.3.15 DTAInterpolationMethod

3.20.3.15.1 Description Parameter for all comparisons involving a distance-to-agreement (DTA) search. Controls how precisely and how often the space between voxel centres are interpolated to identify the exact position of agreement. There are currently three options: no interpolation ('None'), nearest-neighbour ('NN'), and next-nearest-neighbour ('NNN'). (1) If no interpolation is selected, the agreement position will only be established to within approximately the reference image voxels dimensions. To avoid interpolation, voxels that straddle the target value are taken as the agreement distance. Conceptually, if you view a voxel as having a finite spatial extent then this method may be sufficient for distance assessment. Though it is not precise, it is fast. This method will tend to over-estimate the actual distance, though it is possible that it slightly under-estimates it. This method works best when the reference image grid size is small in comparison to the desired spatial accuracy (e.g., if computing gamma, the tolerance should be much larger than the largest voxel dimension) so supersampling is recommended. (2) Nearest-neighbour interpolation considers the line connecting directly adjacent voxels. Using linear interpolation along this line when adjacent voxels straddle the target value, the 3D point where the target value appears can be predicted. This method can significantly improve distance estimation accuracy, though will typically be much slower than no interpolation. On the other hand, this method lower amounts of supersampling, though it is most reliable when the reference image grid size is small in comparison to the desired spatial accuracy. Note that nearest-neighbour interpolation also makes use of the 'no interpolation' methods. If you have a fine reference image, prefer

either no interpolation or nearest-neighbour interpolation. (3) Finally, next-nearest-neighbour considers the diagonally-adjacent neighbours separated by taxi-cab distance of 2 (so in-plane diagonals are considered, but 3D diagonals are not). Quadratic (i.e., bi-linear) interpolation is analytically solved to determine where along the straddling diagonal the target value appears. This method is more expensive than linear interpolation but will generally result in more accurate distance estimates. This method may require lower amounts of supersampling than linear interpolation, but is most reliable when the reference image grid size is small in comparison to the desired spatial accuracy. Use of this method may not be appropriate in all cases considering that supersampling may be needed and a quadratic equation is solved for every voxel diagonal. Note that next-nearest-neighbour interpolation also makes use of the nearest-neighbour and 'no interpolation' methods.

3.20.3.15.2 Default

- `"NN"`

3.20.3.15.3 Supported Options

- `"None"`
- `"NN"`
- `"NNN"`

3.20.3.16 GammaDTAThreshold

3.20.3.16.1 Description Parameter for gamma-index comparisons. Maximally acceptable distance-to-agreement (in DICOM units: mm). When the measured DTA is above this value, the gamma index will necessarily be greater than one. Note this parameter can differ from the DTA_max search cut-off, but should be <= to it.

3.20.3.16.2 Default

- `"5.0"`

3.20.3.16.3 Examples

- `"3.0"`
- `"5.0"`
- `"10.0"`

3.20.3.17 GammaDiscThreshold

3.20.3.17.1 Description Parameter for gamma-index comparisons. Voxel value discrepancies lower than this value are considered acceptable, but values above will result in gamma values >1. The specific interpretation of this parameter (and the units) depend on the specific type of discrepancy used. For percentage-based discrepancies, this parameter is interpretted as a percentage (i.e., '5.0' = '5%'). For voxel intensity measures such as the absolute difference, this value is interpretted as an absolute threshold with the same intensity units (i.e., '5.0' = '5 HU' or similar).

3.20.3.17.2 Default

- `"5.0"`

3.20.3.17.3 Examples

- `"3.0"`
- `"5.0"`
- `"10.0"`

3.20.3.18 GammaTerminateAboveOne

3.20.3.18.1 Description Parameter for gamma-index comparisons. Halt spatial searching if the gamma index will necessarily indicate failure (i.e., gamma >1). Note this can parameter can drastically reduce the computational effort required to compute the gamma index, but the reported gamma values will be invalid whenever they are >1. This is often tolerable since the magnitude only matters when it is <1. In lieu of the true gamma-index, a value slightly >1 will be assumed.

3.20.3.18.2 Default

- `"true"`

3.20.3.18.3 Examples

- `"true"`
- `"false"`

3.21 ContourBasedRayCastDoseAccumulate

3.21.1 Description

This operation performs ray-casting to estimate the dose of a surface. The surface is represented as a set of contours (i.e., an ROI).

3.21.2 Parameters

- DoseLengthMapFileName
- LengthMapFileName
- NormalizedROILabelRegex
- ROILabelRegex
- CylinderRadius
- RaydL
- Rows
- Columns

3.21.2.1 DoseLengthMapFileName

3.21.2.1.1 Description A filename (or full path) for the (dose)*(length traveled through the ROI peel) image map. The format is TBD. Leave empty to dump to generate a unique temporary file.

3.21.2.1.2 Default

- ""

3.21.2.1.3 Examples

- ""
- "/tmp/somefile"
- "localfile.img"
- "derivative_data.img"

3.21.2.2 LengthMapFileName

3.21.2.2.1 Description A filename (or full path) for the (length traveled through the ROI peel) image map. The format is TBD. Leave empty to dump to generate a unique temporary file.

3.21.2.2.2 Default

- ""

3.21.2.2.3 Examples

- ""
- "/tmp/somefile"
- "localfile.img"
- "derivative_data.img"

3.21.2.3 NormalizedROILabelRegex

3.21.2.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.21.2.3.2 Default

- `".*"`

3.21.2.3.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.21.2.4 ROILabelRegex

3.21.2.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.21.2.4.2 Default

- `".*"`

3.21.2.4.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.21.2.5 CylinderRadius

3.21.2.5.1 Description The radius of the cylinder surrounding contour line segments that defines the 'surface'. Quantity is in the DICOM coordinate system.

3.21.2.5.2 Default

- "3.0"

3.21.2.5.3 Examples

- "1.0"
- "2.0"
- "0.5"
- "5.0"

3.21.2.6 RaydL

3.21.2.6.1 Description The distance to move a ray each iteration. Should be « img_thickness and « cylinder_radius. Making too large will invalidate results, causing rays to pass through the surface without registering any dose accumulation. Making too small will cause the run-time to grow and may eventually lead to truncation or round-off errors. Quantity is in the DICOM coordinate system.

3.21.2.6.2 Default

- "0.1"

3.21.2.6.3 Examples

- "0.1"
- "0.05"
- "0.01"
- "0.005"

3.21.2.7 Rows

3.21.2.7.1 Description

The number of rows in the resulting images.

3.21.2.7.2 Default

- "256"

3.21.2.7.3 Examples

- "10"
- "50"
- "128"
- "1024"

3.21.2.8 Columns

3.21.2.8.1 Description

The number of columns in the resulting images.

3.21.2.8.2 Default

- "256"

3.21.2.8.3 Examples

- "10"
- "50"
- "128"
- "1024"

3.22 ContourBooleanOperations

3.22.1 Description

This routine performs 2D Boolean operations on user-provided sets of ROIs. The ROIs themselves are planar contours embedded in R^3, but the Boolean operation is performed once for each 2D plane where the selected ROIs reside. This routine can only perform Boolean operations on co-planar contours. This routine can operate on single contours (rather than ROIs composed of several contours) by simply presenting this routine with a single contour to select.

3.22.2 Notes

- Contour ROI regex matches comprise the sets 'A' and 'B', as in f(A,B) where f is the Boolean operation.
- This routine DOES support disconnected ROIs, such as left- and right-parotid contours that have been joined into a single 'parotids' ROI.

- Many Boolean operations can produce contours with holes. This operation currently connects the interior and exterior with a seam so that holes can be represented by a single polygon (rather than a separate hole polygon). It *is* possible to export holes as contours with a negative orientation, but this was not needed when writing.
- Only the common metadata between contours is propagated to the product contours.

3.22.3 Parameters

- NormalizedROILabelRegexA
- ROILabelRegexA
- NormalizedROILabelRegexB
- ROILabelRegexB
- Operation
- OutputROILabel

3.22.3.1 NormalizedROILabelRegexA

3.22.3.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.22.3.1.2 Default

- `".*"`

3.22.3.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.22.3.2 ROILabelRegexA

3.22.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.22.3.2.2 Default

- `".*"`

3.22.3.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.22.3.3 NormalizedROILabelRegexB

3.22.3.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.22.3.3.2 Default

- `".*"`

3.22.3.3.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.22.3.4 ROILabelRegexB

3.22.3.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.22.3.4.2 Default

- ".*"

3.22.3.4.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.22.3.5 Operation

3.22.3.5.1 Description The Boolean operation (e.g., the function 'f') to perform on the sets of contour polygons 'A' and 'B'. 'Symmetric difference' is also known as 'XOR'.

3.22.3.5.2 Default

- "join"

3.22.3.5.3 Supported Options

- `"intersection"`
- `"join"`
- `"difference"`
- `"symmetric_difference"`

3.22.3.6 OutputROILabel

3.22.3.6.1 Description The label to attach to the ROI contour product of f(A,B).

3.22.3.6.2 Default

- `"Boolean_result"`

3.22.3.6.3 Examples

- `"A+B"`
- `"A-B"`
- `"AuB"`
- `"AnB"`
- `"AxB"`
- `"A^B"`
- `"union"`
- `"xor"`
- `"combined"`
- `"body_without_spinal_cord"`

3.23 ContourSimilarity

3.23.1 Description

This operation estimates the similarity or overlap between two sets of contours. The comparison is based on point samples. It is useful for comparing contouring styles. This operation currently reports Dice and Jaccard similarity metrics.

3.23.2 Notes

- This routine requires an image grid, which is used to control where the contours are sampled. Images are not modified.

3.23.3 Parameters

- ImageSelection
- ROILabelRegexA

- NormalizedROILabelRegexA
- ROILabelRegexB
- NormalizedROILabelRegexB
- FileName
- UserComment

3.23.3.1 ImageSelection

3.23.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.23.3.1.2 Default

- `"last"`

3.23.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`

- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.23.3.2 ROILabelRegexA

3.23.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.23.3.2.2 Default

- `".*"`

3.23.3.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.23.3.3 NormalizedROILabelRegexA

3.23.3.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.23.3.3.2 Default

- ".*"

3.23.3.3.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.23.3.4 ROILabelRegexB

3.23.3.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.23.3.4.2 Default

- ".*"

3.23.3.4.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.23.3.5 NormalizedROILabelRegexB

3.23.3.5.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI

name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.23.3.5.2 Default

- `".*"`

3.23.3.5.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.23.3.6 FileName

3.23.3.6.1 Description A filename (or full path) in which to append similarity data generated by this routine. The format is CSV. Leave empty to dump to generate a unique temporary file.

3.23.3.6.2 Default

- `""`

3.23.3.6.3 Examples

- `""`
- `"/tmp/somefile"`
- `"localfile.csv"`
- `"derivative_data.csv"`

3.23.3.7 UserComment

3.23.3.7.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data.

3.23.3.7.2 Default

- `""`

3.23.3.7.3 Examples

- `""`
- `"Using XYZ"`
- `"Patient treatment plan C"`

3.24 ContourViaGeometry

3.24.1 Description

This operation constructs ROI contours using geometrical primitives.

3.24.2 Notes

- This routine requires an image array onto which the contours will be written.
- This routine expects images to be non-overlapping. In other words, if images overlap then the contours generated may also overlap. This is probably not what you want (but there is nothing intrinsically wrong with presenting this routine with multiple images if you intentionally want overlapping contours).
- Existing contours are ignored and unaltered.
- Small and degenerate contours produced by this routine are suppressed. If a specific number of contours must be generated, provide a slightly larger radius to compensate for the degenerate cases at the extrema.

3.24.3 Parameters

- ROILabel
- ImageSelection
- Shapes

3.24.3.1 ROILabel

3.24.3.1.1 Description A label to attach to the ROI contours.

3.24.3.1.2 Default

- `"unspecified"`

3.24.3.1.3 Examples

- `"unspecified"`
- `"body"`
- `"air"`
- `"bone"`
- `"invalid"`
- `"above_zero"`
- `"below_5.3"`

3.24.3.2 ImageSelection

3.24.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.24.3.2.2 Default

- `"last"`

3.24.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`

- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.24.3.3 Shapes

3.24.3.3.1 Description This parameter is used to specify the shapes to consider. There is currently a single supported shape: sphere. However, it is likely that more shapes will be accepted in the future. Spheres have two configurable parameters: centre and radius. A sphere with centre (1.0,2.0,3.0) and radius 12.3 can be specified as 'sphere(1.0, 2.0, 3.0, 12.3)'.

3.24.3.3.2 Default

- ""

3.24.3.3.3 Examples

- "sphere(-1.0, 2.0, 3.0, 12.3)"

3.25 ContourViaThreshold

3.25.1 Description

This operation constructs ROI contours using images and pixel/voxel value thresholds. There are three methods of contour generation available: a simple binary method in which voxels are either fully in or fully out of the contour, marching squares (which uses linear interpolation to give smooth contours), and a method based on 3D marching cubes that will also provide smooth contours. The marching cubes method does **not** construct a full surface mesh; rather each individual image slice has their own mesh constructed in parallel.

3.25.2 Notes

- This routine expects images to be non-overlapping. In other words, if images overlap then the contours generated may also overlap. This is probably not what you want (but there is nothing intrinsically wrong with presenting this routine with multiple images if you intentionally want overlapping contours).
- Existing contours are ignored and unaltered.

- Contour orientation is (likely) not properly handled in this routine, so 'pinches' and holes will produce contours with inconsistent or invalid topology. If in doubt, disable merge simplifications and live with the computational penalty. The marching cubes approach will properly handle 'pinches' and contours should all be topologically valid.
- Note that the marching-squares method currently only honours the lower threshold.

3.25.3 Parameters

- ROILabel
- Lower
- Upper
- Channel
- ImageSelection
- Method
- SimplifyMergeAdjacent

3.25.3.1 ROILabel

3.25.3.1.1 Description A label to attach to the ROI contours.

3.25.3.1.2 Default

- `"unspecified"`

3.25.3.1.3 Examples

- `"unspecified"`
- `"body"`
- `"air"`
- `"bone"`
- `"invalid"`
- `"above_zero"`
- `"below_5.3"`

3.25.3.2 Lower

3.25.3.2.1 Description The lower bound (inclusive). Pixels with values < this number are excluded from the ROI. If the number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If the number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Both percentages and percentiles are assessed per image array. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile).

3.25.3.2.2 Default

- `"-inf"`

3.25.3.2.3 Examples

- `"0.0"`
- `"-1E-99"`
- `"1.23"`
- `"0.2%"`
- `"23tile"`
- `"23.123 tile"`

3.25.3.3 Upper

3.25.3.3.1 Description The upper bound (inclusive). Pixels with values > this number are excluded from the ROI. If the number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If the number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Both percentages and percentiles are assessed per image array. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile).

3.25.3.3.2 Default

- `"inf"`

3.25.3.3.3 Examples

- `"1.0"`
- `"1E-99"`
- `"2.34"`
- `"98.12%"`
- `"94tile"`
- `"94.123 tile"`

3.25.3.4 Channel

3.25.3.4.1 Description The image channel to use. Zero-based.

3.25.3.4.2 Default

- `"0"`

3.25.3.4.3 Examples

- `"0"`
- `"1"`
- `"2"`

3.25.3.5 ImageSelection

3.25.3.5.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.25.3.5.2 Default

- `"last"`

3.25.3.5.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`

- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.25.3.6 Method

3.25.3.6.1 Description There are currently three supported methods for generating contours: (1) a simple (and fast) 'binary' inclusivity checker, that simply checks if a voxel is within the ROI by testing the value at the voxel centre, (2) the 'marching-squares' method, which samples the corners of every voxel and uses linear interpolation to estimate the threshold value isoline crossings, and (3) a robust but slow method based on 'marching-cubes'. The binary method is fast, but produces extremely jagged contours. It may also have problems with 'pinches' and topological consistency. Marching-squares is reasonably fast and general-purpose, and should produce good quality contours that approximate the threshold value isocurves to first-order. It also handles boundaries well by inserting an extra virtual row and column around the image to ensure contours are all closed. The marching-cubes method is more robust and should reliably produce contours for even the most complicated topologies, but is considerably slower than the binary method. It may produce worse on boundaries, though otherwise it should produce the same contours as marching-squares.

3.25.3.6.2 Default

- `"binary"`

3.25.3.6.3 Supported Options

- `"binary"`
- `"marching-squares"`
- `"marching-cubes"`

3.25.3.7 SimplifyMergeAdjacent

3.25.3.7.1 Description Simplify contours by merging adjacent contours. This reduces the number of contours dramatically, but will cause issues if there are holes (two contours are generated if there is a single hole, but most DICOMautomaton code disregards orientation – so the pixels within the hole will be considered part of the ROI, possibly even doubly so depending on the algorithm). Disabling merges is always safe (and is therefore the default) but can be extremely costly for large images. Furthermore, if you know the ROI does not have holes (or if you don't care) then it is safe to enable merges.

3.25.3.7.2 Default

- `"false"`

3.25.3.7.3 Examples

- `"true"`
- `"false"`

3.26 ContourVote

3.26.1 Description

This routine pits contours against one another using various criteria. A number of 'closest' or 'best' or 'winning' contours are copied into a new contour collection with the specified ROILabel. The original ROIs are not altered, even the winning ROIs.

3.26.2 Notes

- This operation considers individual contours only at the moment. It could be extended to operate on whole ROIs (i.e., contour_collections), or to perform a separate vote within each ROI. The individual contour approach was taken for relevance in 2D image (e.g., RTIMAGE) analysis.
- This operation currently cannot perform voting on multiple criteria. Several criteria could be specified, but an awkward weighting system would also be needed.

3.26.3 Parameters

- WinnerROILabel
- ROILabelRegex
- NormalizedROILabelRegex
- Area
- Perimeter
- CentroidX
- CentroidY
- CentroidZ
- WinnerCount

3.26.3.1 WinnerROILabel

3.26.3.1.1 Description The ROI label to attach to the winning contour(s). All other metadata remains the same.

3.26.3.1.2 Default

- `"unspecified"`

3.26.3.1.3 Examples

- `"closest"`
- `"best"`
- `"winners"`
- `"best-matches"`

3.26.3.2 ROILabelRegex

3.26.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.26.3.2.2 Default

- `".*"`

3.26.3.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.26.3.3 NormalizedROILabelRegex

3.26.3.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.26.3.3.2 Default

- `".*"`

3.26.3.3.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.26.3.4 Area

3.26.3.4.1 Description If this option is provided with a valid positive number, the contour(s) with an area closest to the specified value is/are retained. Note that the DICOM coordinate space is used. (Supplying the default, NaN, will disable this option.) Note: if several criteria are specified, it is not specified in which order they are considered.

3.26.3.4.2 Default

- `"nan"`

3.26.3.4.3 Examples

- `"nan"`
- `"100.0"`
- `"1000"`
- `"10.23E8"`

3.26.3.5 Perimeter

3.26.3.5.1 Description If this option is provided with a valid positive number, the contour(s) with a perimeter closest to the specified value is/are retained. Note that the DICOM coordinate space is used. (Supplying the default, NaN, will disable this option.) Note: if several criteria are specified, it is not specified in which order they are considered.

3.26.3.5.2 Default

- `"nan"`

3.26.3.5.3 Examples

- `"nan"`
- `"0.0"`
- `"123.456"`
- `"1E6"`

3.26.3.6 CentroidX

3.26.3.6.1 Description If this option is provided with a valid positive number, the contour(s) with a centroid closest to the specified value is/are retained. Note that the DICOM coordinate space is used. (Supplying the default, NaN, will disable this option.) Note: if several criteria are specified, it is not specified in which order they are considered.

3.26.3.6.2 Default

- `"nan"`

3.26.3.6.3 Examples

- `"nan"`
- `"0.0"`
- `"123.456"`
- `"-1E6"`

3.26.3.7 CentroidY

3.26.3.7.1 Description If this option is provided with a valid positive number, the contour(s) with a centroid closest to the specified value is/are retained. Note that the DICOM coordinate space is used. (Supplying the default, NaN, will disable this option.) Note: if several criteria are specified, it is not specified in which order they are considered.

3.26.3.7.2 Default

- `"nan"`

3.26.3.7.3 Examples

- `"nan"`
- `"0.0"`
- `"123.456"`
- `"-1E6"`

3.26.3.8 CentroidZ

3.26.3.8.1 Description If this option is provided with a valid positive number, the contour(s) with a centroid closest to the specified value is/are retained. Note that the DICOM coordinate space is used. (Supplying the default, NaN, will disable this option.) Note: if several criteria are specified, it is not specified in which order they are considered.

3.26.3.8.2 Default

- `"nan"`

3.26.3.8.3 Examples

- `"nan"`
- `"0.0"`
- `"123.456"`
- `"-1E6"`

3.26.3.9 WinnerCount

3.26.3.9.1 Description Retain this number of 'best' or 'winning' contours.

3.26.3.9.2 Default

- `"1"`

3.26.3.9.3 Examples

- `"0"`
- `"1"`
- `"3"`
- `"10000"`

3.27 ContourWholeImages

3.27.1 Description

This operation constructs contours for an ROI that encompasses the whole of all specified images. It is useful for operations that operate on ROIs whenever you want to compute something over the whole image. This routine avoids having to manually contour anything. The output is 'ephemeral' and is not commited to any database.

3.27.2 Notes

- This routine will attempt to avoid repeat contours. Generated contours are tested for intersection with an image before the image is processed.
- Existing contours are ignored and unaltered.
- Contours are set slightly inside the outer boundary so they can be easily visualized by overlaying on the image. All voxel centres will be within the bounds.

3.27.3 Parameters

- ROILabel
- ImageSelection

3.27.3.1 ROILabel

3.27.3.1.1 Description A label to attach to the ROI contours.

3.27.3.1.2 Default

- `"everything"`

3.27.3.1.3 Examples

- `"everything"`
- `"whole_images"`
- `"unspecified"`

3.27.3.2 ImageSelection

3.27.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.27.3.2.2 Default

- `"last"`

3.27.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.28 ContouringAides

3.28.1 Description

This operation attempts to prepare an image for easier contouring.

3.28.2 Notes

- At the moment, only logarithmic scaling is applied.

3.28.3 Parameters

No registered options.

3.29 ConvertContoursToMeshes

3.29.1 Description

This routine creates a mesh directly from contours, finding a correspondence between adjacent contours and 'zippering' them together. Please note that this operation is not robust and should only be expected to work for simple, sphere-like contours (i.e., convex polyhedra and mostly-convex polyhedra with only small concavities; see notes for additional information). This operation, when it can be used appropriately, should be significantly faster than meshing via voxelization (e.g., marching cubes). It will also insert no additional vertices on the original contour planes.

3.29.2 Notes

- This routine is experimental and currently relies on simple heuristics to find an adjacent contour correspondence.
- Meshes sliced on the same planes as the original contours *should* reproduce the original contours (barring numerical instabilities). In between the original slices, the mesh may exhibit distortions or obviously invalid correspondence with adjacent contours.
- Mesh 'pairing' on adjacent slices is evaluated using a mutual overlap heuristic. The following adjacent slice pairing scenarios are supported: 1-0, 1-1, N-0, N-1, and N-M (for any N and M greater than 1). Adjacent contours with inconsistent orientations will either be reordered or wholly disregarded. For N-0, N-1, and N-M pairings all contours in N (and M) are fused using with a simple distance heuristic; the fusion bridges are extended off the original contour plane so that mesh slicing will recover the original contours. For 1-0 and N-0 pairings the 'hole' is filled by placing a single vertex offset from the occupied contour plane from the centroid and connecting all vertices; mesh slicing should also recover the original contours in this case.
- Overlapping contours **on the same plane** are **not** currently supported. Only the contour with the largest area will be retained.
- This routine should only be expected to work for simple, sphere-like geometries (i.e., convex polyhedra). Some concavities can be tolerated, but not all. For example, tori can only be meshed if the 'hole' is oriented away from the contour normal. (Otherwise the 'hole' produces concentric contours – which are not supported.) Contours representing convex polyhedra **should** result in manifold meshes, though they may not be watertight and if contour vertices are degenerate (or too close together numerically) meshes will fail to remain manifold.

3.29.3 Parameters

- NormalizedROILabelRegex
- ROILabelRegex
- MeshLabel

3.29.3.1 NormalizedROILabelRegex

3.29.3.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.29.3.1.2 Default

- `".*"`

3.29.3.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.29.3.2 ROILabelRegex

3.29.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.29.3.2.2 Default

- `".*"`

3.29.3.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.29.3.3 MeshLabel

3.29.3.3.1 Description A label to attach to the surface mesh.

3.29.3.3.2 Default

- `"unspecified"`

3.29.3.3.3 Examples

- `"unspecified"`
- `"body"`
- `"air"`
- `"bone"`
- `"invalid"`
- `"above_zero"`
- `"below_5.3"`

3.30 ConvertContoursToPoints

3.30.1 Description

This operation extracts vertices from the selected contours and converts them into a point cloud. Contours are not modified.

3.30.2 Notes

- Existing point clouds are ignored and unaltered.

3.30.3 Parameters

- NormalizedROILabelRegex
- ROILabelRegex
- Label
- Method

3.30.3.1 NormalizedROILabelRegex

3.30.3.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.30.3.1.2 Default

- `".*"`

3.30.3.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.30.3.2 ROILabelRegex

3.30.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.30.3.2.2 Default

- `".*"`

3.30.3.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.30.3.3 Label

3.30.3.3.1 Description A label to attach to the point cloud.

3.30.3.3.2 Default

- `"unspecified"`

3.30.3.3.3 Examples

- `"unspecified"`
- `"POIs"`
- `"peaks"`
- `"above_zero"`
- `"below_5.3"`

3.30.3.4 Method

3.30.3.4.1 Description The conversion method to use. Two options are available: 'vertices' and 'centroid'. The 'vertices' option extracts all vertices from all selected contours and directly inserts them into the new point cloud. Point clouds created this way will contain as many points as there are contour vertices. The 'centroid' option finds the centroid of all vertices from all selected contours. Note that the centroid gives every point an equal weighting, so heterogeneous contour vertex density will shift the position of the centroid (unless the distribution is symmetric about the centroid, which should roughly be the case for spherical contour collections). Point clouds created this way will contain a single point.

3.30.3.4.2 Default

- `"vertices"`

3.30.3.4.3 Supported Options

- `"vertices"`
- `"centroid"`

3.31 ConvertDoseToImage

3.31.1 Description

This operation converts all loaded images from RTDOSE modality to CT modality. Image contents will not change, but the intent to treat as an image or dose matrix will of course change.

3.31.2 Parameters

- Modality

3.31.2.1 Modality

3.31.2.1.1 Description The modality that will replace 'RTDOSE'.

3.31.2.1.2 Default

- `"CT"`

3.31.2.1.3 Examples

- `"CT"`
- `"MR"`
- `"UNKNOWN"`

3.32 ConvertImageToDose

3.32.1 Description

This operation converts all loaded image modalities into RTDOSE. Image contents will not change, but the intent to treat as an image or dose matrix will of course change.

3.32.2 Parameters

No registered options.

3.33 ConvertImageToMeshes

3.33.1 Description

This operation extracts surface meshes from images and pixel/voxel value thresholds. Meshes are appended to the back of the Surface_Mesh stack. There are two methods of contour generation available: a simple binary method in which voxels are either fully in or fully out of the contour, and a method based on marching cubes that will provide smoother contours. Both methods make use of marching cubes – the binary method involves pre-processing.

3.33.2 Notes

- This routine requires images to be regular (i.e., exactly abut nearest adjacent images without any overlap).

3.33.3 Parameters

- ImageSelection
- Lower
- Upper
- Channel
- Method
- MeshLabel

3.33.3.1 ImageSelection

3.33.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.33.3.1.2 Default

- `"last"`

3.33.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.33.3.2 Lower

3.33.3.2.1 Description The lower bound (inclusive). Pixels with values < this number are excluded from the ROI. If the number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If the number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile). Note that computed bounds (i.e., percentages and percentiles) consider the entire image volume.

3.33.3.2.2 Default

- `"-inf"`

3.33.3.2.3 Examples

- `"0.0"`
- `"-1E-99"`

- "1.23"
- "0.2%"
- "23tile"
- "23.123 tile"

3.33.3.3 Upper

3.33.3.3.1 Description

The upper bound (inclusive). Pixels with values > this number are excluded from the ROI. If the number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If the number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile). Note that computed bounds (i.e., percentages and percentiles) consider the entire image volume.

3.33.3.3.2 Default

- "inf"

3.33.3.3.3 Examples

- "1.0"
- "1E-99"
- "2.34"
- "98.12%"
- "94tile"
- "94.123 tile"

3.33.3.4 Channel

3.33.3.4.1 Description

The image channel to use. Zero-based.

3.33.3.4.2 Default

- "0"

3.33.3.4.3 Examples

- "0"
- "1"
- "2"

3.33.3.5 Method

3.33.3.5.1 Description There are currently two supported methods for generating contours: (1) a simple (and fast) binary inclusivity checker, that simply checks if a voxel is within the ROI by testing the value at the voxel centre, and (2) a robust (but slow) method based on marching cubes. The binary method is fast, but produces extremely jagged contours. It may also have problems with 'pinches' and topological consistency. The marching method is more robust and should reliably produce contours for even the most complicated topologies, but is considerably slower than the binary method.

3.33.3.5.2 Default

- `"marching"`

3.33.3.5.3 Supported Options

- `"binary"`
- `"marching"`

3.33.3.6 MeshLabel

3.33.3.6.1 Description A label to attach to the surface mesh.

3.33.3.6.2 Default

- `"unspecified"`

3.33.3.6.3 Examples

- `"unspecified"`
- `"body"`
- `"air"`
- `"bone"`
- `"invalid"`
- `"above_zero"`
- `"below_5.3"`

3.34 ConvertMeshesToContours

3.34.1 Description

This operation constructs ROI contours by slicing the given meshes on a set of image planes.

3.34.2 Notes

- Surface meshes should represent polyhedra.
- This routine does **not** require images to be regular, rectilinear, or even contiguous.
- Images and meshes are unaltered. Existing contours are ignored and unaltered.
- Contour orientation is (likely) not guaranteed to be consistent in this routine.

3.34.3 Parameters

- ROILabel
- MeshSelection
- ImageSelection

3.34.3.1 ROILabel

3.34.3.1.1 Description A label to attach to the ROI contours.

3.34.3.1.2 Default

- `"unspecified"`

3.34.3.1.3 Examples

- `"unspecified"`
- `"body"`
- `"air"`
- `"bone"`
- `"invalid"`
- `"above_zero"`
- `"below_5.3"`

3.34.3.2 MeshSelection

3.34.3.2.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based

indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.34.3.2.2 Default

- `"last"`

3.34.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.34.3.3 ImageSelection

3.34.3.3.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.34.3.3.2 Default

- `"last"`

3.34.3.3.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.35 ConvertMeshesToPoints

3.35.1 Description

This operation converts meshes to point clouds.

3.35.2 Notes

- Meshes are unaltered. Existing point clouds are ignored and unaltered.

3.35.3 Parameters

- MeshSelection
- Label

- Method
- RandomSeed
- RandomSampleDensity

3.35.3.1 MeshSelection

3.35.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.35.3.1.2 Default

- `"last"`

3.35.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`

- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.35.3.2 Label

3.35.3.2.1 Description A label to attach to the point cloud.

3.35.3.2.2 Default

- `"unspecified"`

3.35.3.2.3 Examples

- `"unspecified"`
- `"POIs"`
- `"peaks"`
- `"above_zero"`
- `"below_5.3"`

3.35.3.3 Method

3.35.3.3.1 Description The conversion method to use. Two options are currently available: 'vertices' and 'random'. The 'vertices' option extracts all vertices from all selected meshes and directly inserts them into the new point cloud. Point clouds created this way will contain as many points as there are mesh vertices. The 'random' option samples the surface mesh uniformly. The likelihood of specific a face being sampled is proportional to its area. This method requires a target sample density, which determines the number of samples taken; this density is an average over the entire mesh surface area, and individual samples may have less or more separation from neighbouring samples. Note that the 'random' method will tend to result in clusters of samples and pockets without samples. This is unavoidable when sampling randomly. The 'random' method accepts two parameters: a pseudo-random number generator seed and the desired sample density.

3.35.3.3.2 Default

- `"vertices"`

3.35.3.3.3 Supported Options

- `"vertices"`
- `"random"`

3.35.3.4 RandomSeed

3.35.3.4.1 Description A parameter for the 'random' method: the seed used for the random surface sampling method.

3.35.3.4.2 Default

- `"1595813"`

3.35.3.4.3 Examples

- `"25633"`
- `"20771"`
- `"271"`
- `"1006003"`
- `"11"`
- `"3511"`

3.35.3.5 RandomSampleDensity

3.35.3.5.1 Description A parameter for the 'random' method: the target sample density (as samples/area where area is in DICOM units, nominally mm^{-2})). This parameter effectively controls the total number of samples. Note that the sample density is averaged over the entire surface, so individual samples may cluster or spread out and develop pockets.

3.35.3.5.2 Default

- `"1.0"`

3.35.3.5.3 Examples

- `"0.1"`
- `"0.5"`
- `"1.0"`
- `"5.0"`
- `"10.0"`

3.36 ConvertNaNsToAir

3.36.1 Description

This operation runs the data through a per-pixel filter, converting NaN's to air in Hounsfield units (-1024).

3.36.2 Parameters

No registered options.

3.37 ConvertNaNsToZeros

3.37.1 Description

This operation runs the data through a per-pixel filter, converting NaN's to zeros.

3.37.2 Parameters

No registered options.

3.38 ConvertPixelsToPoints

3.38.1 Description

This operation extracts pixels from the selected images and converts them into a point cloud. Images are not modified.

3.38.2 Notes

- Existing point clouds are ignored and unaltered.

3.38.3 Parameters

- Label
- Lower
- Upper
- Channel
- ImageSelection

3.38.3.1 Label

3.38.3.1.1 Description A label to attach to the point cloud.

3.38.3.1.2 Default

- `"unspecified"`

3.38.3.1.3 Examples

- `"unspecified"`
- `"POIs"`
- `"peaks"`
- `"above_zero"`
- `"below_5.3"`

3.38.3.2 Lower

3.38.3.2.1 Description The lower bound (inclusive). Pixels with values < this number are excluded from the ROI. If the number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If the number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile).

3.38.3.2.2 Default

- `"-inf"`

3.38.3.2.3 Examples

- `"0.0"`
- `"-1E-99"`
- `"1.23"`
- `"0.2%"`
- `"23tile"`
- `"23.123 tile"`

3.38.3.3 Upper

3.38.3.3.1 Description The upper bound (inclusive). Pixels with values > this number are excluded from the ROI. If the number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If the number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile).

3.38.3.3.2 Default

- `"inf"`

3.38.3.3.3 Examples

- `"1.0"`
- `"1E-99"`

- `"2.34"`
- `"98.12%"`
- `"94tile"`
- `"94.123 tile"`

3.38.3.4 Channel

3.38.3.4.1 Description The image channel to use. Zero-based.

3.38.3.4.2 Default

- `"0"`

3.38.3.4.3 Examples

- `"0"`
- `"1"`
- `"2"`

3.38.3.5 ImageSelection

3.38.3.5.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.38.3.5.2 Default

- `"last"`

3.38.3.5.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.39 ConvolveImages

3.39.1 Description

This routine convolves, correlates, or pattern-matches one rectilinear image array with another in voxel number space (i.e., the DICOM coordinate system of the convolution kernel image is entirely disregarded).

3.39.2 Notes

- Both provided image arrays must be rectilinear. In many instances they should both be regular, not just rectilinear, but rectilinearity is sufficient for constructing voxel-by-voxel adjacency relatively quickly, and some applications may require rectilinear kernels to be supported, so rectilinear inputs are permitted.
- This operation can be used to apply arbitrary convolution kernels to an image array. It can also be used to search for instances of one image array in another.
- If the magnitude of the outgoing voxels will be interpretted in absolute (i.e., thresholding based on an absolute magnitude) then the kernel should be weighted so that the sum of all kernel voxel intensities is zero. This will maintain the average voxel intensity. However, for pattern matching the kernel need not be normalized (though it may make interpretting partial matches easier.)

3.39.3 Parameters

- ImageSelection
- ReferenceImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Channel
- Operation

3.39.3.1 ImageSelection

3.39.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.39.3.1.2 Default

- `"last"`

3.39.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`

- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.39.3.2 ReferenceImageSelection

3.39.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.39.3.2.2 Default

- `"last"`

3.39.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`

- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.39.3.3 NormalizedROILabelRegex

3.39.3.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.39.3.3.2 Default

- ".*"

3.39.3.3.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.39.3.4 ROILabelRegex

3.39.3.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.39.3.4.2 Default

- ".*"

3.39.3.4.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.39.3.5 Channel

3.39.3.5.1 Description The channel to operate on (zero-based). Negative values will cause all channels to be operated on.

3.39.3.5.2 Default

- "0"

3.39.3.5.3 Examples

- "-1"
- "0"
- "1"

3.39.3.6 Operation

3.39.3.6.1 Description Controls the way the kernel is applied and the reduction is tallied. Currently, 'convolution', 'correlation', and 'pattern-match' are supported. For convolution, the reference image is spatially inverted along row-, column-, and image-axes. The outgoing voxel intensity is the inner (i.e., dot) product of the paired intensities of the surrounding voxel neighbourhood (i.e., the voxel at (-1,3,0) from the centre of the kernel is paired with the neighbouring voxel at (-1,3,0) from the current/outgoing voxel). For pattern-matching, the difference between the kernel and each voxel's neighbourhood voxels is compared using a 2-norm (i.e., Euclidean) cost function. With this cost function, a perfect, pixel-for-pixel match (i.e., if the kernel images appears exactly in the image being transformed) will result in the outgoing voxel having zero intensity (i.e., zero cost). For correlation, the kernel is applied as-is (just like pattern-matching), but the inner product of the paired voxel neighbourhood intensities is reported (just like convolution). In all cases the kernel is (approximately) centred.

3.39.3.6.2 Default

- `"convolution"`

3.39.3.6.3 Supported Options

- `"convolution"`
- `"correlation"`
- `"pattern-match"`

3.40 CopyContours

3.40.1 Description

This operation deep-copies the selected contours.

3.40.2 Parameters

- NormalizedROILabelRegex
- ROILabelRegex
- ROILabel

3.40.2.1 NormalizedROILabelRegex

3.40.2.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.40.2.1.2 Default

- `".*"`

3.40.2.1.3 Examples

- `".*"`
- `".*Body.*"`

- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.40.2.2 ROILabelRegex

3.40.2.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.40.2.2.2 Default

- ".*"

3.40.2.2.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.40.2.3 ROILabel

3.40.2.3.1 Description A label to attach to the copied ROI contours.

3.40.2.3.2 Default

- "unspecified"

3.40.2.3.3 Examples

- "unspecified"
- "copy"
- "duplicate"
- "bone"

- `"roi_xyz"`

3.41 CopyImages

3.41.1 Description

This operation deep-copies the selected image arrays.

3.41.2 Parameters

- ImageSelection

3.41.2.1 ImageSelection

3.41.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.41.2.1.2 Default

- `"last"`

3.41.2.1.3 Examples

- `"last"`

- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.42 CopyMeshes

3.42.1 Description

This operation deep-copies the selected surface meshes.

3.42.2 Parameters

- MeshSelection

3.42.2.1 MeshSelection

3.42.2.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.42.2.1.2 Default

- `"last"`

3.42.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.43 CopyPoints

3.43.1 Description

This operation deep-copies the selected point clouds.

3.43.2 Parameters

- PointSelection

3.43.2.1 PointSelection

3.43.2.1.1 Description Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex

logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.43.2.1.2 Default

- `"last"`

3.43.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.44 CountVoxels

3.44.1 Description

This operation counts the number of voxels confined to one or more ROIs within a user-provided range.

3.44.2 Notes

- This operation is read-only.

3.44.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex

- Inclusivity
- ContourOverlap
- Lower
- Upper
- Channel
- ResultsSummaryFileName
- UserComment

3.44.3.1 ImageSelection

3.44.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.44.3.1.2 Default

- `"last"`

3.44.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`

- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.44.3.2 NormalizedROILabelRegex

3.44.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.44.3.2.2 Default

- `".*"`

3.44.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.44.3.3 ROILabelRegex

3.44.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.44.3.3.2 Default

- ".*"

3.44.3.3.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.44.3.4 Inclusivity

3.44.3.4.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.44.3.4.2 Default

- "center"

3.44.3.4.3 Supported Options

- "center"
- "centre"
- "planar_corner_inclusive"
- "planar_inc"
- "planar_corner_exclusive"
- "planar_exc"

3.44.3.5 ContourOverlap

3.44.3.5.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant

for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.44.3.5.2 Default

- `"ignore"`

3.44.3.5.3 Supported Options

- `"ignore"`
- `"honour_opposite_orientations"`
- `"overlapping_contours_cancel"`
- `"honour_opps"`
- `"overlap_cancel"`

3.44.3.6 Lower

3.44.3.6.1 Description The lower bound (inclusive). Pixels with values < this number are excluded from the ROI. If the number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If the number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile).

3.44.3.6.2 Default

- `"-inf"`

3.44.3.6.3 Examples

- `"0.0"`
- `"-1E-99"`
- `"1.23"`
- `"0.2%"`
- `"23tile"`
- `"23.123 tile"`

3.44.3.7 Upper

3.44.3.7.1 Description The upper bound (inclusive). Pixels with values > this number are excluded from the ROI. If the number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If the number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile).

3.44.3.7.2 Default

- "inf"

3.44.3.7.3 Examples

- "1.0"
- "1E-99"
- "2.34"
- "98.12%"
- "94tile"
- "94.123 tile"

3.44.3.8 Channel

3.44.3.8.1 Description The image channel to use. Zero-based.

3.44.3.8.2 Default

- "0"

3.44.3.8.3 Examples

- "0"
- "1"
- "2"

3.44.3.9 ResultsSummaryFileName

3.44.3.9.1 Description This file will contain a brief summary of the results. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.44.3.9.2 Default

- ""

3.44.3.9.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.44.3.10 UserComment

3.44.3.10.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data.

3.44.3.10.2 Default

- `""`

3.44.3.10.3 Examples

- `""`
- `"Using XYZ"`
- `"Patient treatment plan C"`

3.45 CropImageDoseToROIs

3.45.1 Description

This operation crops image slices to the specified ROI(s), with an additional margin.

3.45.2 Parameters

- DICOMMargin
- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex

3.45.2.1 DICOMMargin

3.45.2.1.1 Description The amount of margin (in the DICOM coordinate system) to surround the ROI(s).

3.45.2.1.2 Default

- `"0.5"`

3.45.2.1.3 Examples

- `"0.1"`
- `"2.0"`
- `"-0.5"`
- `"20.0"`

3.45.2.2 ImageSelection

3.45.2.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.45.2.2.2 Default

- `"last"`

3.45.2.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.45.2.3 NormalizedROILabelRegex

3.45.2.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.45.2.3.2 Default

- `".*"`

3.45.2.3.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.45.2.4 ROILabelRegex

3.45.2.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.45.2.4.2 Default

- `".*"`

3.45.2.4.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`

- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.46 CropImages

3.46.1 Description

This operation crops image slices in either pixel or DICOM coordinate spaces.

3.46.2 Parameters

- ImageSelection
- RowsL
- RowsH
- ColumnsL
- ColumnsH
- DICOMMargin

3.46.2.1 ImageSelection

3.46.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.46.2.1.2 Default

- `"all"`

3.46.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.46.2.2 RowsL

3.46.2.2.1 Description The number of rows to remove, starting with the first row. Can be absolute (px), percentage (%), or distance in terms of the DICOM coordinate system. Note the DICOM coordinate system can be flipped, so the first row can be either on the top or bottom of the image.

3.46.2.2.2 Default

- `"0px"`

3.46.2.2.3 Examples

- `"0px"`
- `"10px"`
- `"100px"`
- `"15%"`
- `"15.75%"`
- `"123.45"`

3.46.2.3 RowsH

3.46.2.3.1 Description The number of rows to remove, starting with the last row. Can be absolute (px), percentage (%), or distance in terms of the DICOM coordinate system. Note the DICOM coordinate system can be flipped, so the first row can be either on the top or bottom of the image.

3.46.2.3.2 Default

- `"0px"`

3.46.2.3.3 Examples

- `"0px"`
- `"10px"`
- `"100px"`
- `"15%"`
- `"15.75%"`
- `"123.45"`

3.46.2.4 ColumnsL

3.46.2.4.1 Description The number of columns to remove, starting with the first column. Can be absolute (px), percentage (%), or distance in terms of the DICOM coordinate system. Note the DICOM coordinate system can be flipped, so the first column can be either on the top or bottom of the image.

3.46.2.4.2 Default

- `"0px"`

3.46.2.4.3 Examples

- `"0px"`
- `"10px"`
- `"100px"`
- `"15%"`
- `"15.75%"`
- `"123.45"`

3.46.2.5 ColumnsH

3.46.2.5.1 Description The number of columns to remove, starting with the last column. Can be absolute (px), percentage (%), or distance in terms of the DICOM coordinate system. Note the DICOM coordinate system can be flipped, so the first column can be either on the top or bottom of the image.

3.46.2.5.2 Default

- `"0px"`

3.46.2.5.3 Examples

- `"0px"`
- `"10px"`
- `"100px"`
- `"15%"`
- `"15.75%"`
- `"123.45"`

3.46.2.6 DICOMMargin

3.46.2.6.1 Description The amount of margin (in the DICOM coordinate system) to spare from cropping.

3.46.2.6.2 Default

- `"0.0"`

3.46.2.6.3 Examples

- `"0.1"`
- `"2.0"`
- `"-0.5"`
- `"20.0"`

3.47 CropROIDose

3.47.1 Description

This operation provides a simplified interface for overriding voxel values outside a ROI. For example, this operation can be used to modify a base plan by eliminating dose outside an OAR.

3.47.2 Notes

- This operation performs the opposite of the 'Trim' operation, which trims voxel values **inside** a ROI.
- The inclusivity of a voxel that straddles the ROI boundary can be specified in various ways. Refer to the Inclusivity parameter documentation.

3.47.3 Parameters

- Channel
- ImageSelection
- ContourOverlap
- Inclusivity

- Method
- ExteriorVal
- InteriorVal
- ExteriorOverwrite
- InteriorOverwrite
- NormalizedROILabelRegex
- ROILabelRegex
- ImageSelection
- Filename
- ParanoiaLevel

3.47.3.1 Channel

3.47.3.1.1 Description The image channel to use. Zero-based. Use '-1' to operate on all available channels.

3.47.3.1.2 Default

- `"-1"`

3.47.3.1.3 Examples

- `"-1"`
- `"0"`
- `"1"`
- `"2"`

3.47.3.2 ImageSelection

3.47.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of

sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.47.3.2.2 Default

- `"all"`

3.47.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.47.3.3 ContourOverlap

3.47.3.3.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. This will effectively honour only the outermost contour regardless of orientation, but provides the most predictable and consistent results. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. This is useful for Boolean structures where contour orientation is significant for interior contours (holes). If contours do not have consistent overlap (e.g., if contours intersect) the results can be unpredictable and hard to interpret. The option 'overlapping_contours_cancel' ignores orientation and alternately cancerls all overlapping contours. Again, if the contours do not have consistent overlap (e.g., if contours intersect) the results can be unpredictable and hard to interpret.

3.47.3.3.2 Default

- `"ignore"`

3.47.3.3.3 Supported Options

- `"ignore"`
- `"honour_opposite_orientations"`
- `"overlapping_contours_cancel"`
- `"honour_opps"`
- `"overlap_cancel"`

3.47.3.4 Inclusivity

3.47.3.4.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.47.3.4.2 Default

- `"planar_inc"`

3.47.3.4.3 Supported Options

- `"center"`
- `"centre"`
- `"planar_corner_inclusive"`
- `"planar_inc"`
- `"planar_corner_exclusive"`
- `"planar_exc"`

3.47.3.5 Method

3.47.3.5.1 Description Controls the type of image mask that is generated. The default, 'binary', exclusively overwrites voxels with the InteriorValue or ExteriorValue. Another method is 'receding_squares' which creates a mask which, if processed with the marching-squares algorithm, will (mostly) recreate the original contours. The 'receding_squares' can be considered the inverse of the marching-squares algorithm. Note that the 'receding_squares' implementation is not optimized for speed.

3.47.3.5.2 Default

- `"binary"`

3.47.3.5.3 Supported Options

- "binary"
- "receding_squares"

3.47.3.6 ExteriorVal

3.47.3.6.1 Description The value to give to voxels outside the specified ROI(s). For the 'binary' method, note that this value will be ignored if exterior overwrites are disabled. For the 'receding_squares' method this value is used to define the threshold needed to recover the original contours (mean of InteriorVal and ExteriorVal).

3.47.3.6.2 Default

- "0.0"

3.47.3.6.3 Examples

- "0.0"
- "-1.0"
- "1.23"
- "2.34E26"

3.47.3.7 InteriorVal

3.47.3.7.1 Description The value to give to voxels within the specified ROI(s). For the 'binary' method, note that this value will be ignored if interior overwrites are disabled. For the 'receding_squares' method this value is used to define the threshold needed to recover the original contours (mean of InteriorVal and ExteriorVal).

3.47.3.7.2 Default

- "0.0"

3.47.3.7.3 Examples

- "0.0"
- "-1.0"
- "1.23"
- "2.34E26"

3.47.3.8 ExteriorOverwrite

3.47.3.8.1 Description Whether to overwrite voxels exterior to the specified ROI(s).

3.47.3.8.2 Default

- `"true"`

3.47.3.8.3 Examples

- `"true"`
- `"false"`

3.47.3.9 InteriorOverwrite

3.47.3.9.1 Description Whether to overwrite voxels interior to the specified ROI(s).

3.47.3.9.2 Default

- `"false"`

3.47.3.9.3 Examples

- `"true"`
- `"false"`

3.47.3.10 NormalizedROILabelRegex

3.47.3.10.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.47.3.10.2 Default

- `".*"`

3.47.3.10.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.47.3.11 ROILabelRegex

3.47.3.11.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.47.3.11.2 Default

- ".*"

3.47.3.11.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.47.3.12 ImageSelection

3.47.3.12.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based

indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.47.3.12.2 Default

- `"all"`

3.47.3.12.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.47.3.13 Filename

3.47.3.13.1 Description The filename (or full path name) to which the DICOM file should be written.

3.47.3.13.2 Default

- `"/tmp/RD.dcm"`

3.47.3.13.3 Examples

- `"/tmp/RD.dcm"`

- "./RD.dcm"
- "RD.dcm"

3.47.3.14 ParanoiaLevel

3.47.3.14.1 Description At low paranoia setting, only top-level UIDs are replaced. At medium paranoia setting, many UIDs, descriptions, and labels are replaced, but the PatientID and FrameOfReferenceUID are retained. The high paranoia setting is the same as the medium setting, but the PatientID and FrameOfReferenceUID are also replaced. (Note: this is not a full anonymization.) Use the low setting if you want to retain linkage to the originating data set. Use the medium setting if you don't. Use the high setting if your TPS goes overboard linking data sets by PatientID and/or FrameOfReferenceUID.

3.47.3.14.2 Default

- "medium"

3.47.3.14.3 Supported Options

- "low"
- "medium"
- "high"

3.48 DCEMRI_IAUC

3.48.1 Description

This operation will compute the Integrated Area Under the Curve (IAUC) for any images present.

3.48.2 Notes

- This operation is not optimized in any way and operates on whole images. It can be fairly slow, especially if the image volume is huge, so it is best to crop images if possible.

3.48.3 Parameters

No registered options.

3.49 DCEMRI_Nonparametric_CE

3.49.1 Description

This operation takes a single DCE-MRI scan ('measurement') and generates a "poor-mans's" contrast enhancement signal. This is accomplished by subtracting the pre-contrast injection images average ('baseline') from later images (and then possibly/optionally averaging relative to the baseline).

3.49.2 Notes

- Only a single image volume is required. It is expected to have temporal sampling beyond the contrast injection timepoint (or some default value – currently around ~30s). The resulting images retain the baseline portion, so you'll need to trim yourself if needed.
- Be aware that this method of deriving contrast enhancement is not valid! It ignores nuances due to differing T1 or T2 values due to the presence of contrast agent. It should only be used for exploratory purposes or cases where the distinction with reality is irrelevant.

3.49.3 Parameters

No registered options.

3.50 DICOMExportContours

3.50.1 Description

This operation exports the selected contours to a DICOM RTSTRUCT-modality file.

3.50.2 Notes

- There are various 'paranoia' levels that can be used to partially anonymize the output. In particular, most metadata and UIDs are replaced, but the files may still be recognized by a determined individual by comparing the contour data. Do NOT rely on this routine to fully anonymize the data!

3.50.3 Parameters

- Filename
- ParanoiaLevel
- NormalizedROILabelRegex
- ROILabelRegex

3.50.3.1 Filename

3.50.3.1.1 Description The filename (or full path name) to which the DICOM file should be written.

3.50.3.1.2 Default

- `"/tmp/RTSTRUCT.dcm"`

3.50.3.1.3 Examples

- `"/tmp/RTSTRUCT.dcm"`
- `"./RTSTRUCT.dcm"`
- `"RTSTRUCT.dcm"`

3.50.3.2 ParanoiaLevel

3.50.3.2.1 Description At low paranoia setting, only top-level UIDs are replaced. At medium paranoia setting, many UIDs, descriptions, and labels are replaced, but the PatientID and FrameOfReferenceUID are retained. The high paranoia setting is the same as the medium setting, but the PatientID and FrameOfReferenceUID are also replaced. (Note: this is not a full anonymization.) Use the low setting if you want to retain linkage to the originating data set. Use the medium setting if you don't. Use the high setting if your TPS goes overboard linking data sets by PatientID and/or FrameOfReferenceUID.

3.50.3.2.2 Default

- `"medium"`

3.50.3.2.3 Supported Options

- `"low"`
- `"medium"`
- `"high"`

3.50.3.3 NormalizedROILabelRegex

3.50.3.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful

for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.50.3.3.2 Default

- `".*"`

3.50.3.3.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.50.3.4 ROILabelRegex

3.50.3.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.50.3.4.2 Default

- `".*"`

3.50.3.4.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.51 DICOMExportImagesAsCT

3.51.1 Description

This operation exports the selected Image_Array(s) to DICOM CT-modality files.

3.51.2 Notes

- There are various 'paranoia' levels that can be used to partially anonymize the output. In particular, most metadata and UIDs are replaced, but the files may still be recognized by a determined individual by comparing the coordinate system and pixel values. Do NOT rely on this routine to fully anonymize the data!

3.51.3 Parameters

- ImageSelection
- Filename
- ParanoiaLevel

3.51.3.1 ImageSelection

3.51.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.51.3.1.2 Default

- `"last"`

3.51.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.51.3.2 Filename

3.51.3.2.1 Description The filename (or full path name) to which the DICOM files should be written. The file format is a gzipped-TAR file containing multiple CT-modality files.

3.51.3.2.2 Default

- `"CTs.tgz"`

3.51.3.2.3 Examples

- `"/tmp/CTs.tgz"`
- `"./CTs.tar.gz"`
- `"CTs.tgz"`

3.51.3.3 ParanoiaLevel

3.51.3.3.1 Description At low paranoia setting, only top-level UIDs are replaced. At medium paranoia setting, many UIDs, descriptions, and labels are replaced, but the PatientID and FrameOfReferenceUID are retained. The high paranoia setting is the same as the medium setting, but the PatientID and FrameOfReferenceUID are also replaced. (Note: this is not a full anonymization.) Use the low setting if you want to retain linkage to the originating data set. Use the medium setting if you don't. Use the high setting if your TPS goes overboard linking data sets by PatientID and/or FrameOfReferenceUID.

3.51.3.3.2 Default

- `"medium"`

3.51.3.3.3 Supported Options

- `"low"`
- `"medium"`
- `"high"`

3.52 DICOMExportImagesAsDose

3.52.1 Description

This operation exports the selected Image_Array to a DICOM dose file.

3.52.2 Notes

- There are various 'paranoia' levels that can be used to partially anonymize the output. In particular, most metadata and UIDs are replaced, but the files may still be recognized by a determined individual by comparing the coordinate system and pixel values. Do NOT rely on this routine to fully anonymize the data!

3.52.3 Parameters

- ImageSelection
- Filename
- ParanoiaLevel

3.52.3.1 ImageSelection

3.52.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.52.3.1.2 Default

- `"last"`

3.52.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.52.3.2 Filename

3.52.3.2.1 Description The filename (or full path name) to which the DICOM file should be written.

3.52.3.2.2 Default

- `"/tmp/RD.dcm"`

3.52.3.2.3 Examples

- `"/tmp/RD.dcm"`
- `"./RD.dcm"`
- `"RD.dcm"`

3.52.3.3 ParanoiaLevel

3.52.3.3.1 Description At low paranoia setting, only top-level UIDs are replaced. At medium paranoia setting, many UIDs, descriptions, and labels are replaced, but the PatientID and FrameOfReferenceUID are retained. The high paranoia setting is the same as the medium setting, but the PatientID and FrameOfReferenceUID are also replaced. (Note: this is not a full anonymization.) Use the low setting if you want to retain linkage to the originating data set. Use the medium setting if you don't. Use the high setting if your TPS goes overboard linking data sets by PatientID and/or FrameOfReferenceUID.

3.52.3.3.2 Default

- `"medium"`

3.52.3.3.3 Supported Options

- `"low"`
- `"medium"`
- `"high"`

3.53 DeDuplicateImages

3.53.1 Description

This operation de-duplicates image arrays, identifying sets of duplicates based on user-specified criteria and purging all but one of the duplicates.

3.53.2 Notes

- This routine is experimental.

3.53.3 Parameters

- ImageSelection

3.53.3.1 ImageSelection

3.53.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.53.3.1.2 Default

- `"all"`

3.53.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.54 DecayDoseOverTimeHalve

3.54.1 Description

This operation transforms a dose map (assumed to be delivered some distant time in the past) to simulate 'decay' or 'evaporation' or 'forgivance' of radiation dose by simply halving the value. This model is only appropriate at long time-scales, but there is no cut-off or threshold to denote what is sufficiently 'long'. So use at your own risk. As a rule of thumb, do not use this routine if fewer than 2-3y have elapsed.

3.54.2 Notes

- This routine will combine spatially-overlapping images by summing voxel intensities. So if you have a time course it may be more sensible to aggregate images in some way (e.g., spatial averaging) prior to calling this routine.
- Since this routine is meant to be applied multiple times in succession for different ROIs (which possibly overlap), all images are imbued with a second channel that is treated as a mask. Mask channels are permanently attached so that multiple passes will not erroneously decay dose. If this will be problematic, the extra column should be trimmed immediately after calling this routine.

3.54.3 Parameters

- NormalizedROILabelRegex
- ROILabelRegex

3.54.3.1 NormalizedROILabelRegex

3.54.3.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.54.3.1.2 Default

- `".*"`

3.54.3.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.54.3.2 ROILabelRegex

3.54.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.54.3.2.2 Default

- `".*"`

3.54.3.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.55 DecayDoseOverTimeJones2014

3.55.1 Description

This operation transforms a dose map (delivered some time in the past) to account for tissue recovery (i.e., 'dose decay,' 'dose evaporation,' or 'dose forgivance') using the time-dependent model of Jones and Grant (2014; doi:10.1016/j.clon.2014.04.027). This model is specific to reirradiation of central nervous tissues. See the Jones and Grant paper or 'Nasopharyngeal Carcinoma' by Wai Tong Ng et al. (2016; doi:10.1007/174_2016_48) for more information.

3.55.2 Notes

- This routine will combine spatially-overlapping images by summing voxel intensities. So if you have a time course it may be more sensible to aggregate images in some way (e.g., spatial averaging) prior to calling this routine.

- Since this routine is meant to be applied multiple times in succession for different ROIs (which possibly overlap), all images are imbued with a second channel that is treated as a mask. Mask channels are permanently attached so that multiple passes will not erroneously decay dose. If this will be problematic, the extra column should be trimmed immediately after calling this routine.

3.55.3 Parameters

- NormalizedROILabelRegex
- ROILabelRegex
- Course1NumberOfFractions
- ToleranceTotalDose
- ToleranceNumberOfFractions
- TimeGap
- AlphaBetaRatio
- UseMoreConservativeRecovery

3.55.3.1 NormalizedROILabelRegex

3.55.3.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.55.3.1.2 Default

- `".*"`

3.55.3.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.55.3.2 ROILabelRegex

3.55.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.55.3.2.2 Default

- `".*"`

3.55.3.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.55.3.3 Course1NumberOfFractions

3.55.3.3.1 Description The number of fractions delivered for the first (i.e., previous) course. If several apply, you can provide a single effective fractionation scheme's 'n'.

3.55.3.3.2 Default

- `"35"`

3.55.3.3.3 Examples

- `"15"`
- `"25"`
- `"30.001"`
- `"35.3"`

3.55.3.4 ToleranceTotalDose

3.55.3.4.1 Description The dose delivered (in Gray) for a hypothetical 'lifetime dose tolerance' course. This dose corresponds to a hypothetical radiation course that nominally corresponds to the toxicity of interest. For CNS tissues, it will probably be myelopathy or necrosis at some population-level onset risk (e.g., 5% risk of myelopathy). The value provided will be converted to a BED_{a/b} so you can safely provide a 'nominal' value. Be aware that each voxel is treated independently, rather than treating OARs/ROIs as a whole. (Many dose limits reported in the literature use whole-ROI D_mean or D_max, and so may be not be directly applicable to per-voxel risk estimation!) Note that the QUANTEC 2010 reports almost all assume 2 Gy/fraction. If several fractionation schemes were used, you should provide a cumulative BED-derived dose here.

3.55.3.4.2 Default

- `"52"`

3.55.3.4.3 Examples

- `"15"`
- `"20"`
- `"25"`
- `"50"`
- `"83.2"`

3.55.3.5 ToleranceNumberOfFractions

3.55.3.5.1 Description The number of fractions ('n') for the 'lifetime dose tolerance' toxicity you are interested in. Note that this is converted to a BED_{a/b} so you can safely provide a 'nominal' value. If several apply, you can provide a single effective fractionation scheme's 'n'.

3.55.3.5.2 Default

- `"35"`

3.55.3.5.3 Examples

- `"15"`
- `"25"`
- `"30.001"`
- `"35.3"`

3.55.3.6 TimeGap

3.55.3.6.1 Description The number of years between radiotherapy courses. Note that this is normally estimated by (1) extracting study/series dates from the provided dose files and (2) using the current date as the second course date. Use this parameter to override the autodetected gap time. Note: if the provided value is negative, autodetection will be used. Autodetection can fail if the data has been anonymized with date-shifting.

3.55.3.6.2 Default

- `"-1"`

3.55.3.6.3 Examples

- `"0.91"`
- `"2.6"`
- `"5"`

3.55.3.7 AlphaBetaRatio

3.55.3.7.1 Description The ratio alpha/beta (in Gray) to use when converting to a biologically-equivalent dose distribution for central nervous tissues. Jones and Grant (2014) recommend alpha/beta = 2 Gy to be conservative. It is more commonplace to use alpha/beta = 3 Gy, but this is less conservative and there is some evidence that it may be erroneous to use 3 Gy.

3.55.3.7.2 Default

- `"2"`

3.55.3.7.3 Examples

- `"2"`
- `"2.5"`
- `"3"`

3.55.3.8 UseMoreConservativeRecovery

3.55.3.8.1 Description Jones and Grant (2014) provide two ways to estimate the function 'r'. One is fitted to experimental data, and one is a more conservative estimate of the fitted function. This parameter controls whether or not the more conservative function is used.

3.55.3.8.2 Default

- `"true"`

3.55.3.8.3 Examples

- `"true"`
- `"false"`

3.56 DecimatePixels

3.56.1 Description

This operation spatially aggregates blocks of pixels, thereby decimating them and making the images consume far less memory. The precise size reduction and spatial aggregate can be set in the source.

3.56.2 Parameters

- OutSizeR
- OutSizeC

3.56.2.1 OutSizeR

3.56.2.1.1 Description The number of pixels along the row unit vector to group into an outgoing pixel. Must be a multiplicative factor of the incoming image's row count. No decimation occurs if either this or 'OutSizeC' is zero or negative.

3.56.2.1.2 Default

- `"8"`

3.56.2.1.3 Examples

- `"0"`
- `"2"`
- `"4"`
- `"8"`
- `"16"`
- `"32"`
- `"64"`
- `"128"`
- `"256"`
- `"512"`

3.56.2.2 OutSizeC

3.56.2.2.1 Description The number of pixels along the column unit vector to group into an outgoing pixel. Must be a multiplicative factor of the incoming image's column count. No decimation occurs if either this or 'OutSizeR' is zero or negative.

3.56.2.2.2 Default

- `"8"`

3.56.2.2.3 Examples

- `"0"`
- `"2"`
- `"4"`
- `"8"`
- `"16"`
- `"32"`
- `"64"`
- `"128"`
- `"256"`
- `"512"`

3.57 DeleteContours

3.57.1 Description

This operation deletes the selected contours.

3.57.2 Notes

- Contours can be shallow copies that are shared amongst multiple Drover class instances. Deleting contours in one Drover instance will delete them from all linked instances. Typically, contours are deep-copied to avoid this problem, but be aware if using shallow copies.

3.57.3 Parameters

- NormalizedROILabelRegex
- ROILabelRegex

3.57.3.1 NormalizedROILabelRegex

3.57.3.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.57.3.1.2 Default

- `".*"`

3.57.3.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.57.3.2 ROILabelRegex

3.57.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.57.3.2.2 Default

- `".*"`

3.57.3.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`

- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.58 DeleteImages

3.58.1 Description

This routine deletes images from memory. It is most useful when working with positional operations in stages.

3.58.2 Parameters

- ImageSelection

3.58.2.1 ImageSelection

3.58.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.58.2.1.2 Default

- `"last"`

3.58.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.59 DeleteMeshes

3.59.1 Description

This routine deletes surface meshes from memory. It is most useful when working with positional operations in stages.

3.59.2 Parameters

- MeshSelection

3.59.2.1 MeshSelection

3.59.2.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that

'!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.59.2.1.2 Default

- `"last"`

3.59.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.60 DeletePoints

3.60.1 Description

This routine deletes point clouds from memory. It is most useful when working with positional operations in stages.

3.60.2 Parameters

- PointSelection

3.60.2.1 PointSelection

3.60.2.1.1 Description Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based

indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.60.2.1.2 Default

- `"last"`

3.60.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.61 DetectGrid3D

3.61.1 Description

This routine fits a 3D grid to a point cloud using a Procrustes analysis with point-to-model correspondence estimated via an iterative closest point approach. A RANSAC-powered loop is used to (1) randomly select a subset of the grid for coarse iterative closest point grid fitting, and then (2) use the coarse fit results as a guess for the whole point cloud in a refinement stage.

3.61.2 Notes

- Traditional Procrustes analysis requires a priori point-to-point correspondence knowledge. Because this operation fits a model (with infinite extent), point-to-point correspondence is not known and the model is effectively an infinite continuum of potential points. To overcome this problem, correspondence is estimated by projecting each point in the point cloud onto every grid line and selecting the closest projected point. The point cloud point and the project point are then treated as corresponding points. Using this phony correspondence, the Procrustes problem is solved and the grid is reoriented. This is performed iteratively. However **there is no guarantee the procedure will converge** and furthermore, even if it does converge, **there is no guarantee that the grid will be appropriately fit**. The best results will occur when the grid is already closely aligned with the point cloud (i.e., when the first guess is very close to a solution). If this cannot be guaranteed, it may be advantageous to have a nearly continuous point cloud to avoid gaps in which the iteration can get stuck in a local minimum. For this reason, RANSAC is applied to continuously reboot the fitting procedure. All but the best fit are discarded.

- A two-stage RANSAC inner-loop iterative closest point fitting procedure is used. Coarse grid fitting is first performed with a limited subset of the whole point cloud. This is followed with a refinment stage in which the enire point cloud is fitted using an initial guess carried forward from the coarse fitting stage. This guess is expected to be reasonably close to the true grid in cases where the coarse fitting procedure was not tainted by outliers, but is only derived from a small portion of the point cloud. (Thus RANSAC is used to perform this coarse-fine iterative procedure multiple times to provide resilience to poor-quality coarse fits.) CoarseICPMaxLoops is the maximum number of iterative-closest point loop iterations performed during the coarse grid fitting stage (on a subset of the point cloud), and FineICPMaxLoops is the maximum number of iterative-closest point loop iterations performed during the refinement stage (using the whole point cloud). Note that, depending on the noise level and number of points considered (i.e., whether the RANSACDist parameter is sufficiently small to avoid spatial wrapping of corresponding points into adjacent grid cells, but sufficiently large to enclose at least one whole grid cell), the coarse phase should converge within a few iterations. However, on each loop a single point is selected as the grid's rotation centre. This means that a few extra iterations should always be used in case outliers are selected as rotation centres. Additionally, if the point cloud is dense or there are lots of outliers present, increase CoarseICPMaxLoops to ensure there is a reasonable chance of selecting legitimate rotation points. On the other hand, be aware that the coarse-fine iterative procedure is performed afresh for every RANSAC loop, and RANSAC loops are better able to ensure the point cloud is sampled ergodically. It might therefore be more productive

to increase the RANSACMaxLoops parameter and reduce the number of CoarseICPMaxLoops. FineICPMaxLoops should converge quickly if the coarse fitting stage was representative of the true grid. However, as in the coarse stage a rotation centre is nominated in each loop, so it will be a good idea to keep a sufficient number of loops to ensure a legitimate and appropriate non-outlier point is nominated during this stage. Given the complicated interplay between parameters and stages, it is always best to tune using a representative sample of the point cloud you need to fit!

3.61.3 Parameters

- PointSelection
- GridSeparation
- RANSACDist
- GridSampling
- LineThickness
- RandomSeed
- RANSACMaxLoops
- CoarseICPMaxLoops
- FineICPMaxLoops
- ResultsSummaryFileName
- UserComment

3.61.3.1 PointSelection

3.61.3.1.1 Description Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.61.3.1.2 Default

- `"last"`

3.61.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.61.3.2 GridSeparation

3.61.3.2.1 Description The separation of the grid (in DICOM units; mm) being fit. This parameter describes how close adjacent grid lines are to one another. Separation is measured from one grid line centre to the nearest adjacent grid line centre.

3.61.3.2.2 Default

- `"10.0"`

3.61.3.2.3 Examples

- `"10.0"`
- `"15.5"`
- `"25.0"`
- `"1.23E4"`

3.61.3.3 RANSACDist

3.61.3.3.1 Description Every iteration of RANSAC selects a single point from the point cloud. Only the near-vicinity of points are retained for iterative-closest-point Procrustes solving. This parameter determines the maximum radial distance from the RANSAC point within which point cloud points will be

retained; all points further than this distance away will be pruned for a given round of RANSAC. This is needed because corresponding points begin to alias to incorrect cell faces when the ICP procedure begins with a poor guess. Pruning points in a spherical neighbourhood with a diameter 2-4x the GridSeparation (so a radius 1-2x GridSeparation) will help mitigate aliasing even when the initial guess is poor. However, smaller windows may increase susceptibility to noise/outliers, and RANSACDist should never be smaller than a grid voxel. If RANSACDist is not provided, a default of (1.5 * GridSeparation) is used.

3.61.3.3.2 Default

- `"nan"`

3.61.3.3.3 Examples

- `"7.0"`
- `"10.0"`
- `"2.46E4"`

3.61.3.4 GridSampling

3.61.3.4.1 Description Specifies how the grid data has been sampled. Use value '1' if only grid cell corners (i.e., '0D' grid intersections) are sampled. Use value '2' if grid cell edges (i.e., 1D grid lines) are sampled. Use value '3' if grid cell faces (i.e., 2D planar faces) are sampled.

3.61.3.4.2 Default

- `"1"`

3.61.3.4.3 Examples

- `"1"`
- `"2"`
- `"3"`

3.61.3.5 LineThickness

3.61.3.5.1 Description The thickness of grid lines (in DICOM units; mm). If zero, lines are treated simply as lines. If non-zero, grid lines are treated as hollow cylinders with a diameter of this thickness.

3.61.3.5.2 Default

- `"0.0"`

3.61.3.5.3 Examples

- "1.0"
- "1.5"
- "10.0"
- "1.23E4"

3.61.3.6 RandomSeed

3.61.3.6.1 Description A whole number seed value to use for random number generation.

3.61.3.6.2 Default

- "1317"

3.61.3.6.3 Examples

- "1"
- "2"
- "1113523431"

3.61.3.7 RANSACMaxLoops

3.61.3.7.1 Description The maximum number of iterations of RANSAC. (See operation notes for further details.)

3.61.3.7.2 Default

- "100"

3.61.3.7.3 Examples

- "100"
- "2000"
- "1E4"

3.61.3.8 CoarseICPMaxLoops

3.61.3.8.1 Description Coarse grid fitting is performed with a limited subset of the whole point cloud. This is followed with a refinment stage in which the enire point is fitted using an initial guess from the coarse fitting stage. CoarseICPMaxLoops is the maximum number of iterative-closest point loop iterations performed during the coarse grid fitting stage. (See operation notes for further details.)

3.61.3.8.2 Default

- "10"

3.61.3.8.3 Examples

- "10"
- "100"
- "1E4"

3.61.3.9 FineICPMaxLoops

3.61.3.9.1 Description Coarse grid fitting is performed with a limited subset of the whole point cloud. This is followed with a refinment stage in which the enire point is fitted using an initial guess from the coarse fitting stage. FineICPMaxLoops is the maximum number of iterative-closest point loop iterations performed during the refinement stage. (See operation notes for further details.)

3.61.3.9.2 Default

- "20"

3.61.3.9.3 Examples

- "10"
- "50"
- "100"

3.61.3.10 ResultsSummaryFileName

3.61.3.10.1 Description This file will contain a brief summary of the results. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.61.3.10.2 Default

- ""

3.61.3.10.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.61.3.11 UserComment

3.61.3.11.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data.

3.61.3.11.2 Default

- `""`

3.61.3.11.3 Examples

- `""`
- `"Using XYZ"`
- `"Patient treatment plan C"`

3.62 DetectShapes3D

3.62.1 Description

This operation attempts to detect shapes in image volumes.

3.62.2 Parameters

- ImageSelection

3.62.2.1 ImageSelection

3.62.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.62.2.1.2 Default

- `"all"`

3.62.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.63 DrawGeometry

3.63.1 Description

This operation draws shapes and patterns on images. Drawing is confined to one or more ROIs.

3.63.2 Parameters

- ImageSelection
- VoxelValue
- Overwrite
- Channel
- NormalizedROILabelRegex
- ROILabelRegex
- ContourOverlap
- Inclusivity
- Shapes

3.63.2.1 ImageSelection

3.63.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D

volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.63.2.1.2 Default

- `"last"`

3.63.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.63.2.2 VoxelValue

3.63.2.2.1 Description The value to give voxels which are coincident with a point from the point cloud.

3.63.2.2.2 Default

- `"1.0"`

3.63.2.2.3 Examples

- `"-1.0"`
- `"0.0"`
- `"1.23"`
- `"nan"`
- `"inf"`

3.63.2.3 Overwrite

3.63.2.3.1 Description Whether to overwrite voxels interior or exterior to the specified ROI(s).

3.63.2.3.2 Default

- `"interior"`

3.63.2.3.3 Supported Options

- `"interior"`
- `"exterior"`

3.63.2.4 Channel

3.63.2.4.1 Description The image channel to use. Zero-based.

3.63.2.4.2 Default

- `"0"`

3.63.2.4.3 Examples

- `"0"`
- `"1"`
- `"2"`

3.63.2.5 NormalizedROILabelRegex

3.63.2.5.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.63.2.5.2 Default

- `".*"`

3.63.2.5.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.63.2.6 ROILabelRegex

3.63.2.6.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.63.2.6.2 Default

- `".*"`

3.63.2.6.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.63.2.7 ContourOverlap

3.63.2.7.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.63.2.7.2 Default

- "ignore"

3.63.2.7.3 Supported Options

- "ignore"
- "honour_opposite_orientations"
- "overlapping_contours_cancel"
- "honour_opps"
- "overlap_cancel"

3.63.2.8 Inclusivity

3.63.2.8.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.63.2.8.2 Default

- "center"

3.63.2.8.3 Supported Options

- `"center"`
- `"centre"`
- `"planar_corner_inclusive"`
- `"planar_inc"`
- `"planar_corner_exclusive"`
- `"planar_exc"`

3.63.2.9 Shapes

3.63.2.9.1 Description This parameter is used to specify the shapes and patterns to consider. Currently grids, wireframecubes, and solidspheres are available. Grids have four configurable parameters: two orientation unit vectors, line thickness, and line separation. A grid intersecting at the image array's centre, aligned with (1.0,0.0,0.0) and (0.0,1.0,0.0), with line thickness (i.e., diameter) 3.0 (DICOM units; mm), and separation 15.0 can be specified as 'grid(1.0,0.0,0.0, 0.0,1.0,0.0, 3.0, 15.0)'. Unit vectors will be Gram-Schmidt orthogonalized. Note that currently the grid *must* intersect the image array's centre. Cubes have the same number of configurable parameters, but only a single cube of the grid is drawn. The wireframecube is centred at the image centre, rather than intersecting it. Solid spheres have two configurable parameters: a centre vector and a radius. A solid sphere at (1.0,2.0,3.0) with radius 15.0 (all DICOM units; mm) can be specified as 'solidsphere(1.0,2.0,3.0, 15.0)'. Grid, wireframecube, and solidsphere shapes only overwrite voxels that intersect the geometry (i.e., the surface if hollow or the internal volume if solid) permitting easier composition of multiple shapes or custom backgrounds.

3.63.2.9.2 Default

- `"grid(-0.0941083,0.995562,0, 0.992667,0.0938347,0.0762047, 3.0, 15.0)"`

3.63.2.9.3 Examples

- `"grid(1.0,0.0,0.0, 0.0,1.0,0.0, 3.0, 15.0)"`
- `"wireframecube(1.0,0.0,0.0, 0.0,1.0,0.0, 3.0, 15.0)"`
- `"solidsphere(0.0,0.0,0.0, 15.0)"`

3.64 DroverDebug

3.64.1 Description

This operation reports basic information on the state of the main Drover class. It can be used to report on the state of the data, which can be useful for debugging.

3.64.2 Parameters

- IncludeMetadata

3.64.2.1 IncludeMetadata

3.64.2.1.1 Description Whether to include metadata in the output. This data can significantly increase the size of the output.

3.64.2.1.2 Default

- `"false"`

3.64.2.1.3 Examples

- `"true"`
- `"false"`

3.65 DumpAllOrderedImageMetadataToFile

3.65.1 Description

Dump exactly what order the data will be in for the following analysis.

3.65.2 Parameters

No registered options.

3.66 DumpAnEncompassedPoint

3.66.1 Description

This operation estimates the number of spatially-overlapping images. It finds an arbitrary point within an arbitrary image, and then finds all other images which encompass the point.

3.66.2 Parameters

No registered options.

3.67 DumpFilesPartitionedByTime

3.67.1 Description

This operation prints PACS filenames along with the associated time. It is more focused than the metadata dumpers above. This data can be used for many things, such as image viewers which are not DICOM-aware or deformable registration on time series data.

3.67.2 Parameters

No registered options.

3.68 DumpImageMeshes

3.68.1 Description

This operation exports images as a 3D surface mesh model (structured ASCII Wavefront OBJ) that can be manipulated in various ways (e.g., stereographic projection). Note that the mesh will be a 3D depiction of the image(s) as they naturally are – meshes will always be rectangular. A companion material library file (MTL) assigns colours to each ROI based on the voxel intensity.

3.68.2 Notes

- Each image is processed separately. Each mesh effectively produces a 2D relief map embedded into a 3D model that can be easily rendered to produce various effects (e.g., perspective, stereoscopy, extrusion, surface smoothing, etc.).

3.68.3 Parameters

- ImageSelection
- OutBase
- HistogramBins
- MagnitudeAmplification
- Normalize

3.68.3.1 ImageSelection

3.68.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be ‘first’, ‘last’, ‘none’, or ‘all’ literals. Additionally ‘#N’ for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, ‘#-N’ selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a ‘!’.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a ‘!’). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the ‘numerous’ and ‘fewest’ literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a ‘!’. Note that ‘!numerous’ means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., ‘fewest’).

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ‘;’ and are applied in the order specified.

3.68.3.1.2 Default

- `"all"`

3.68.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.68.3.2 OutBase

3.68.3.2.1 Description A base filename (or full path) in which to (over)write image mesh and material library files. File formats are Wavefront Object (obj) and Material Library (mtl). Every image will receive one unique and sequentially-numbered obj and mtl file using this prefix.

3.68.3.2.2 Default

- `"/tmp/dicomautomaton_dumpimagemeshes_"`

3.68.3.2.3 Examples

- `"/tmp/image_mesh_"`
- `"./"`
- `"../model_"`

3.68.3.3 HistogramBins

3.68.3.3.1 Description The number of equal-width bins pixel intensities should be grouped into. Binning is performed in order to more easily associate material properties with pixels. If pixel intensities were continuous, each pixel would receive its own material definition. This could result in enormous MTL files and wasted disk space. Binning solves this issue. However, if images are small or must be differentiated precisely consider using a large number of bins. Otherwise 150-1000 bins should suffice for display purposes.

3.68.3.3.2 Default

- `"255"`

3.68.3.3.3 Examples

- `"10"`
- `"50"`
- `"100"`
- `"200"`
- `"500"`

3.68.3.4 MagnitudeAmplification

3.68.3.4.1 Description Pixel magnitudes (i.e., intensities) are scaled according to the image thickness, but a small gap is left between meshes so that abutting images do not quite intersect (this can cause non-manifold scenarios). However, if stackability is not a concern then pixel magnitudes can be magnified to exaggerate the relief effect. A value of 1.0 provides no magnification. A value of 2.0 provides 2x magnification, but note that the base of each pixel is slightly offset from the top to avoid top-bottom face intersections, even when magnification is 0.0.

3.68.3.4.2 Default

- `"1.0"`

3.68.3.4.3 Examples

- `"0.75"`
- `"1.0"`

- `"2.0"`
- `"5.0"`
- `"75.6"`

3.68.3.5 Normalize

3.68.3.5.1 Description This parameter controls whether the model will be 'normalized,' which effectively makes the outgoing model more consistent for all images. Currently this means centring the model at (0,0,0), mapping the row and column directions to (1,0,0) and (0,1,0) respectively, and scaling the image (respecting the aspect ratio) to fit within a bounding square of size 100x100 (DICOM units; mm). If normalization is *not* used, the image mesh will inherit the spatial characteristics of the image it is derived from.

3.68.3.5.2 Default

- `"false"`

3.68.3.5.3 Examples

- `"true"`
- `"false"`

3.69 DumpImageMetadataOccurrencesToFile

3.69.1 Description

Dump all the metadata elements, but group like-items together and also print the occurence number.

3.69.2 Parameters

- ImageSelection
- FileName
- UserComment

3.69.2.1 ImageSelection

3.69.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based

indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.69.2.1.2 Default

- `"last"`

3.69.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.69.2.2 FileName

3.69.2.2.1 Description A filename (or full path) in which to append metadata reported by this routine. The format is tab-separated values (TSV). Leave empty to dump to generate a unique temporary file.

3.69.2.2.2 Default

- `""`

3.69.2.2.3 Examples

- ""
- "/tmp/somefile"
- "localfile.tsv"
- "derivative_data.tsv"

3.69.2.3 UserComment

3.69.2.3.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. If left empty, the column will be empty in the output.

3.69.2.3.2 Default

- ""

3.69.2.3.3 Examples

- ""
- "Using XYZ"
- "Patient treatment plan C"

3.70 DumpPerROIParams_KineticModel_1C2I_5P

3.70.1 Description

Given a perfusion model, this routine computes parameter estimates for ROIs.

3.70.2 Parameters

- ROILabelRegex
- Filename
- Separator

3.70.2.1 ROILabelRegex

3.70.2.1.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single)

regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.70.2.1.2 Default

- `".*"`

3.70.2.1.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.70.2.2 Filename

3.70.2.2.1 Description A file into which the results should be dumped. If the filename is empty, the results are dumped to the console only.

3.70.2.2.2 Default

- `""`

3.70.2.2.3 Examples

- `"/tmp/results.txt"`
- `"/dev/null"`
- `"~/output.txt"`

3.70.2.3 Separator

3.70.2.3.1 Description The token(s) to place between adjacent columns of output. Note: because whitespace is trimmed from user parameters, whitespace separators other than the default are shortened to an empty string. So non-default whitespace are not currently supported.

3.70.2.3.2 Default

- `" "`

3.70.2.3.3 Examples

- ","
- ";"
- "_a_long_separator_"

3.71 DumpPixelValuesOverTimeForAnEncompassedPoint

3.71.1 Description

Output the pixel values over time for a generic point. Currently the point is arbitrarily taken to tbe the centre of the first image. This is useful for quickly and programmatically inspecting trends, but the SFML_Viewer operation is better for interactive exploration.

3.71.2 Parameters

No registered options.

3.72 DumpPlanSummary

3.72.1 Description

This operation dumps a summary of a radiotherapy plan. This operation can be used to gain insight into a plan from a high-level overview.

3.72.2 Parameters

- SummaryFileName
- UserComment

3.72.2.1 SummaryFileName

3.72.2.1.1 Description A filename (or full path) in which to append summary data generated by this routine. The format is CSV. Leave empty to dump to generate a unique temporary file.

3.72.2.1.2 Default

- ""

3.72.2.1.3 Examples

- ""
- "/tmp/somefile"

- "localfile.csv"
- "derivative_data.csv"

3.72.2.2 UserComment

3.72.2.2.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. If left empty, the column will be omitted from the output.

3.72.2.2.2 Default

- ""

3.72.2.2.3 Examples

- ""
- "Using XYZ"
- "Patient treatment plan C"

3.73 DumpROIContours

3.73.1 Description

This operation exports contours in a standard surface mesh format (structured ASCII Wavefront OBJ) in planar polygon format. A companion material library file (MTL) assigns colours to each ROI to help differentiate them.

3.73.2 Notes

- Contours that are grouped together into a contour_collection are treated as a logical within the output. For example, all contours in a collection will share a common material property (e.g., colour). If more fine-grained grouping is required, this routine can be called once for each group which will result in a logical grouping of one ROI per file.

3.73.3 Parameters

- DumpFileName
- MTLFileName
- NormalizedROILabelRegex
- ROILabelRegex

3.73.3.1 DumpFileName

3.73.3.1.1 Description A filename (or full path) in which to (over)write with contour data. File format is Wavefront obj. Leave empty to dump to generate a unique temporary file.

3.73.3.1.2 Default

- ""

3.73.3.1.3 Examples

- ""
- "/tmp/somefile.obj"
- "localfile.obj"
- "derivative_data.obj"

3.73.3.2 MTLFileName

3.73.3.2.1 Description A filename (or full path) in which to (over)write a Wavefront material library file. This file is used to colour the contours to help differentiate them. Leave empty to dump to generate a unique temporary file.

3.73.3.2.2 Default

- ""

3.73.3.2.3 Examples

- ""
- "/tmp/materials.mtl"
- "localfile.mtl"
- "somefile.mtl"

3.73.3.3 NormalizedROILabelRegex

3.73.3.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.73.3.3.2 Default

- ".*"

3.73.3.3.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.73.3.4 ROILabelRegex

3.73.3.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.73.3.4.2 Default

- ".*"

3.73.3.4.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.74 DumpROIData

3.74.1 Description

This operation dumps ROI contour information for debugging and quick inspection purposes.

3.74.2 Parameters

No registered options.

3.75 DumpROISNR

3.75.1 Description

This operation computes the Signal-to-Noise ratio (SNR) for each ROI. The specific 'SNR' computed is SNR = (mean pixel) / (pixel std dev) which is the inverse of the coefficient of variation.

3.75.2 Notes

- This routine will combine spatially-overlapping images by summing voxel intensities. So if you have a time course it may be more sensible to aggregate images in some way (e.g., spatial averaging) prior to calling this routine.

3.75.3 Parameters

- SNRFileName
- NormalizedROILabelRegex
- ROILabelRegex

3.75.3.1 SNRFileName

3.75.3.1.1 Description A filename (or full path) in which to append SNR data generated by this routine. The format is CSV. Leave empty to dump to generate a unique temporary file.

3.75.3.1.2 Default

- `""`

3.75.3.1.3 Examples

- `""`
- `"/tmp/somefile"`
- `"localfile.csv"`
- `"derivative_data.csv"`

3.75.3.2 NormalizedROILabelRegex

3.75.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.75.3.2.2 Default

- `".*"`

3.75.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.75.3.3 ROILabelRegex

3.75.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.75.3.3.2 Default

- `".*"`

3.75.3.3.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.76 DumpROISurfaceMeshes

3.76.1 Description

This operation generates surface meshes from contour volumes. Output is written to file(s) for viewing with an external viewer (e.g., meshlab).

3.76.2 Notes

- This routine is currently limited. Many parameters can only be modified via recompilation. This will be addressed in a future version.

3.76.3 Parameters

- OutBase
- NormalizedROILabelRegex
- ROILabelRegex
- GridRows
- GridColumns
- ContourOverlap
- Inclusivity

3.76.3.1 OutBase

3.76.3.1.1 Description The prefix of the filename that surface mesh files will be saved as. If no name is given, unique names will be chosen automatically.

3.76.3.1.2 Default

- ""

3.76.3.1.3 Examples

- "/tmp/dicomautomaton_dumproisurfacemesh"
- "../somedir/output"
- "/path/to/some/mesh"

3.76.3.2 NormalizedROILabelRegex

3.76.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.76.3.2.2 Default

- `".*"`

3.76.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.76.3.3 ROILabelRegex

3.76.3.3.1 Description A regex matching ROI labels/names to consider. The default will match all available ROIs. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regex to avoid them. All ROIs have to match the single regex, so use the 'or' token if needed. Regex is case insensitive and uses grep syntax.

3.76.3.3.2 Default

- `".*"`

3.76.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`

- "Gross_Liver"
- ".*parotid.*|.*sub.*mand.*"
- "left_parotid|right_parotid|eyes"

3.76.3.4 GridRows

3.76.3.4.1 Description Controls the spatial resolution of the grid used to approximate the ROI(s). Specifically, the number of rows. Note that the number of slices is fixed by the contour separation. A larger number will result in a more accurate mesh, but will also result longer runtimes and higher mesh complexity. Setting this parameter too high will result in excessive runtime and memory usage, so consider post-processing (i.e., subdivision) if a smooth mesh is needed.

3.76.3.4.2 Default

- "256"

3.76.3.4.3 Examples

- "64"
- "128"
- "256"
- "512"
- "1024"

3.76.3.5 GridColumns

3.76.3.5.1 Description Controls the spatial resolution of the grid used to approximate the ROI(s). (Refer to GridRows for more information.)

3.76.3.5.2 Default

- "256"

3.76.3.5.3 Examples

- "64"
- "128"
- "256"
- "512"
- "1024"

3.76.3.6 ContourOverlap

3.76.3.6.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.76.3.6.2 Default

- `"ignore"`

3.76.3.6.3 Supported Options

- `"ignore"`
- `"honour_opposite_orientations"`
- `"overlapping_contours_cancel"`
- `"honour_opps"`
- `"overlap_cancel"`

3.76.3.7 Inclusivity

3.76.3.7.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.76.3.7.2 Default

- `"center"`

3.76.3.7.3 Supported Options

- `"center"`
- `"centre"`
- `"planar_corner_inclusive"`
- `"planar_inc"`
- `"planar_corner_exclusive"`
- `"planar_exc"`

3.77 DumpTPlanMetadataOccurrencesToFile

3.77.1 Description

Dump all the metadata elements, but group like-items together and also print the occurence number.

3.77.2 Parameters

- TPlanSelection
- FileName
- UserComment

3.77.2.1 TPlanSelection

3.77.2.1.1 Description Select one or more treatment plans. Note that a single treatment plan may be composed of multiple beams; if delivered sequentially, they should collectively represent a single logically cohesive plan. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth treatment plan (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last treatment plan. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the treatment plan composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all treatment plan that do not have the greatest number of sub-objects, not the least-numerous treatment plan (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.77.2.1.2 Default

- `"last"`

3.77.2.1.3 Examples

- `"last"`
- `"first"`

- "all"
- "none"
- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.77.2.2 FileName

3.77.2.2.1 Description A filename (or full path) in which to append metadata reported by this routine. The format is tab-separated values (TSV). Leave empty to dump to generate a unique temporary file.

3.77.2.2.2 Default

- ""

3.77.2.2.3 Examples

- ""
- "/tmp/somefile"
- "localfile.tsv"
- "derivative_data.tsv"

3.77.2.3 UserComment

3.77.2.3.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. If left empty, the column will be empty in the output.

3.77.2.3.2 Default

- ""

3.77.2.3.3 Examples

- ""
- "Using XYZ"
- "Patient treatment plan C"

3.78 DumpVoxelDoseInfo

3.78.1 Description

This operation locates the minimum and maximum dose voxel values. It is useful for estimating prescription doses.

3.78.2 Notes

- This implementation makes use of a primitive way of estimating dose. Please verify it works (or re-write using the new methods) before using for anything important.

3.78.3 Parameters

No registered options.

3.79 EvaluateDoseVolumeStats

3.79.1 Description

This operation evaluates a variety of Dose-Volume statistics. It is geared toward PTV ROIs. Currently the following are implemented: (1) Dose Homogeneity Index: H = (D_{2%} - D_{98%})/D_{median} | over one or more PTVs, where D_{2%} is the maximum dose that covers 2% of the volume of the PTV, and D_{98%} is the minimum dose that covers 98% of the volume of the PTV. (2) Conformity Number: C = V_{T,pres}ˆ{2} / (V_{T} * V_{pres}) where V_{T,pres} is the PTV volume receiving at least 95% of the PTV prescription dose, V_{T} is the volume of the PTV, and V_{pres} is volume of all (tissue) voxels receiving at least 95% of the PTV prescription dose.

3.79.2 Notes

- This routine will combine spatially-overlapping images by summing voxel intensities. It will not combine separate image_arrays though. If needed, you'll have to perform a meld on them beforehand.

3.79.3 Parameters

- OutFileName
- PTVPrescriptionDose
- PTVROILabelRegex
- PTVNormalizedROILabelRegex
- BodyROILabelRegex
- BodyNormalizedROILabelRegex
- UserComment

3.79.3.1 OutFileName

3.79.3.1.1 Description A filename (or full path) in which to append dose statistic data generated by this routine. The format is CSV. Leave empty to dump to generate a unique temporary file.

3.79.3.1.2 Default

- ""

3.79.3.1.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.79.3.2 PTVPrescriptionDose

3.79.3.2.1 Description The dose prescribed to the PTV of interest (in Gy).

3.79.3.2.2 Default

- "70"

3.79.3.2.3 Examples

- "50"
- "66"
- "70"
- "82.5"

3.79.3.3 PTVROILabelRegex

3.79.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.79.3.3.2 Default

- ".*"

3.79.3.3.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.79.3.4 PTVNormalizedROILabelRegex

3.79.3.4.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.79.3.4.2 Default

- ".*"

3.79.3.4.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.79.3.5 BodyROILabelRegex

3.79.3.5.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.79.3.5.2 Default

- `".*"`

3.79.3.5.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.79.3.6 BodyNormalizedROILabelRegex

3.79.3.6.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.79.3.6.2 Default

- `".*"`

3.79.3.6.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.79.3.7 UserComment

3.79.3.7.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. If left empty, the column will be omitted from the output.

3.79.3.7.2 Default

- ""

3.79.3.7.3 Examples

- ""
- "Using XYZ"
- "Patient treatment plan C"

3.80 EvaluateNTCPModels

3.80.1 Description

This operation evaluates a variety of NTCP models for each provided ROI. The selected ROI should be OARs. Currently the following are implemented: (1) The LKB model. (2) The 'Fenwick' model for solid tumours (in the lung; for a whole-lung OAR).

3.80.2 Notes

- Generally these models require dose in 2 Gy per fraction equivalents ('EQD2'). You must pre-convert the data if the RT plan is not already 2 Gy per fraction. There is no easy way to ensure this conversion has taken place or was unnecessary.
- This routine will combine spatially-overlapping images by summing voxel intensities. So if you have a time course it may be more sensible to aggregate images in some way (e.g., spatial averaging) prior to calling this routine.

- The LKB and mEUD both have their own gEUD 'alpha' parameter, but they are not necessarily shared. Huang et al. 2015 (doi:10.1038/srep18010) used alpha=1 for the LKB model and alpha=5 for the mEUD model.

3.80.3 Parameters

- NTCPFileName
- NormalizedROILabelRegex
- ROILabelRegex
- LKB_TD50
- LKB_M
- LKB_Alpha
- UserComment

3.80.3.1 NTCPFileName

3.80.3.1.1 Description A filename (or full path) in which to append NTCP data generated by this routine. The format is CSV. Leave empty to dump to generate a unique temporary file.

3.80.3.1.2 Default

- `""`

3.80.3.1.3 Examples

- `""`
- `"/tmp/somefile"`
- `"localfile.csv"`
- `"derivative_data.csv"`

3.80.3.2 NormalizedROILabelRegex

3.80.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.80.3.2.2 Default

- ".*"

3.80.3.2.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.80.3.3 ROILabelRegex

3.80.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.80.3.3.2 Default

- ".*"

3.80.3.3.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.80.3.4 LKB_TD50

3.80.3.4.1 Description The dose (in Gray) needed to deliver to the selected OAR that will induce the effect in 50% of cases.

3.80.3.4.2 Default

- `"26.8"`

3.80.3.4.3 Examples

- `"26.8"`

3.80.3.5 LKB_M

3.80.3.5.1 Description No description given...

3.80.3.5.2 Default

- `"0.45"`

3.80.3.5.3 Examples

- `"0.45"`

3.80.3.6 LKB_Alpha

3.80.3.6.1 Description The weighting factor α that controls the relative weighting of volume and dose in the generalized Equivalent Uniform Dose (gEUD) model. When $\alpha = 1$, the gEUD is equivalent to the mean; when $\alpha = 0$, the gEUD is equivalent to the geometric mean. Wu et al. (doi:10.1016/S0360-3016(01)02585-8) claim that for normal tissues, α can be related to the Lyman-Kutcher-Burman (LKB) model volume parameter 'n' via $\alpha = 1/n$. Sovik et al. (doi:10.1016/j.ejmp.2007.09.001) found that gEUD is not strongly impacted by errors in α. Niemierko et al. ('A generalized concept of equivalent uniform dose. Med Phys 26:1100, 1999) generated maximum likelihood estimates for 'several tumors and normal structures' which ranged from -13.1 for local control of chordoma tumors to +17.7 for perforation of esophagus. Gay et al. (doi:10.1016/j.ejmp.2007.07.001) table 2 lists estimates based on the work of Emami (doi:10.1016/0360-3016(91)90171-Y) for normal tissues ranging from 1-31. Brenner et al. (doi:10.1016/0360-3016(93)90189-3) recommend -7.2 for breast cancer, -10 for melanoma, and -13 for squamous cell carcinomas. A 2017 presentation by Ontida Apinorasethkul claims the tumour range spans [-40:-1] and the organs at risk range spans [1:40]. AAPM TG report 166 also provides a listing of recommended values, suggesting -10 for PTV and GTV, +1 for parotid, 20 for spinal cord, and 8-16 for rectum, bladder, brainstem, chiasm, eye, and optic nerve. Burman (1991) and QUANTEC (2010) also provide estimates.

3.80.3.6.2 Default

- `"1.0"`

3.80.3.6.3 Examples

- "1"
- "3"
- "4"
- "20"
- "31"

3.80.3.7 UserComment

3.80.3.7.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. If left empty, the column will be omitted from the output.

3.80.3.7.2 Default

- ""

3.80.3.7.3 Examples

- ""
- "Using XYZ"
- "Patient treatment plan C"

3.81 EvaluateTCPModels

3.81.1 Description

This operation evaluates a variety of TCP models for each provided ROI. The selected ROI should be the GTV (according to the Fenwick model). Currently the following are implemented: (1) The 'Martel' model. (2) Equivalent Uniform Dose (EUD) TCP. (3) The 'Fenwick' model for solid tumours.

3.81.2 Notes

- Generally these models require dose in 2Gy/fractions equivalents ('EQD2'). You must pre-convert the data if the RT plan is not already 2Gy/fraction. There is no easy way to ensure this conversion has taken place or was unnecessary.
- This routine will combine spatially-overlapping images by summing voxel intensities. So if you have a time course it may be more sensible to aggregate images in some way (e.g., spatial averaging) prior to calling this routine.

- The Fenwick and Martel models share the value of D_{50}. There may be a slight difference in some cases. Huang et al. 2015 (doi:10.1038/srep18010) used both models and used 84.5 Gy for the Martel model while using 84.6 Gy for the Fenwick model. (The paper also reported using a Fenwick 'm' of 0.329 whereas the original report by Fenwick reported 0.392, so I don't think this should be taken as strong evidence of the equality of D_{50}. However, the difference seems relatively insignificant.)

3.81.3 Parameters

- TCPFileName
- NormalizedROILabelRegex
- ROILabelRegex
- Gamma50
- Dose50
- EUD_Gamma50
- EUD_TCD50
- EUD_Alpha
- Fenwick_C
- Fenwick_M
- Fenwick_Vref
- UserComment

3.81.3.1 TCPFileName

3.81.3.1.1 Description A filename (or full path) in which to append TCP data generated by this routine. The format is CSV. Leave empty to dump to generate a unique temporary file.

3.81.3.1.2 Default

- `""`

3.81.3.1.3 Examples

- `""`
- `"/tmp/somefile"`
- `"localfile.csv"`
- `"derivative_data.csv"`

3.81.3.2 NormalizedROILabelRegex

3.81.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI

name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.81.3.2.2 Default

- `".*"`

3.81.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.81.3.3 ROILabelRegex

3.81.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.81.3.3.2 Default

- `".*"`

3.81.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`

- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.81.3.4 Gamma50

3.81.3.4.1 Description The unitless 'normalized dose-response gradient' or normalized slope of the logistic dose-response model at the half-maximum point (e.g., D_50). Informally, this parameter controls the steepness of the dose-response curve. (For more specific information, consult a standard reference such as 'Basic Clinical Radiobiology' 4th Edition by Joiner et al., sections 5.3-5.5.) This parameter is empirically fit and not universal. Late endpoints for normal tissues have gamma_50 around 2-6 whereas gamma_50 nominally varies around 1.5-2.5 for local control of squamous cell carcinomas of the head and neck.

3.81.3.4.2 Default

- "2.3"

3.81.3.4.3 Examples

- "1.5"
- "2"
- "2.5"
- "6"

3.81.3.5 Dose50

3.81.3.5.1 Description The dose (in Gray) needed to achieve 50% probability of local tumour control according to an empirical logistic dose-response model (e.g., D_50). Informally, this parameter 'shifts' the model along the dose axis. (For more specific information, consult a standard reference such as 'Basic Clinical Radiobiology' 4th Edition by Joiner et al., sections 5.1-5.3.) This parameter is empirically fit and not universal. In 'Quantifying the position and steepness of radiation dose-response curves' by Bentzen and Tucker in 1994, D_50 of around 60-65 Gy are reported for local control of head and neck cancers (pyriform sinus carcinoma and neck nodes with max diameter <= 3cm). Martel et al. report 84.5 Gy in lung.

3.81.3.5.2 Default

- "65"

3.81.3.5.3 Examples

- "37.9"
- "52"

- "60"
- "65"
- "84.5"

3.81.3.6 EUD_Gamma50

3.81.3.6.1 Description The unitless 'normalized dose-response gradient' or normalized slope of the gEUD TCP model. It is defined only for the generalized Equivalent Uniform Dose (gEUD) model. This is sometimes referred to as the change in TCP for a unit change in dose straddled at the TCD_50 dose. It is a counterpart to the Martel model's 'Gamma_50' parameter, but is not quite the same. Okunieff et al. (doi:10.1016/0360-3016(94)00475-Z) computed Gamma50 for tumours in human subjects across multiple institutions; they found a median of 0.8 for gross disease and a median of 1.5 for microscopic disease. The inter-quartile range was [0.7:1.8] and [0.7:2.2] respectively. (Refer to table 3 for site-specific values.) Additionally, Gay et al. (doi:10.1016/j.ejmp.2007.07.001) claim that a value of 4.0 for late effects a value of 2.0 for tumors in 'are reasonable initial estimates in [our] experience.' Their table 2 lists (NTCP) estimates based on the work of Emami (doi:10.1016/0360-3016(91)90171-Y).

3.81.3.6.2 Default

- "0.8"

3.81.3.6.3 Examples

- "0.8"
- "1.5"

3.81.3.7 EUD_TCD50

3.81.3.7.1 Description The uniform dose (in Gray) needed to deliver to the tumour to achieve 50% probability of local control. It is defined only for the generalized Equivalent Uniform Dose (gEUD) model. It is a counterpart to the Martel model's 'Dose_50' parameter, but is not quite the same (n.b., TCD_50 is a uniform dose whereas D_50 is more like a per voxel TCP-weighted mean.) Okunieff et al. (doi:10.1016/0360-3016(94)00475-Z) computed TCD50 for tumours in human subjects across multiple institutions; they found a median of 51.9 Gy for gross disease and a median of 37.9 Gy for microscopic disease. The inter-quartile range was [38.4:62.8] and [27.0:49.1] respectively. (Refer to table 3 for site-specific values.) Gay et al. (doi:10.1016/j.ejmp.2007.07.001) table 2 lists (NTCP) estimates based on the work of Emami (doi:10.1016/0360-3016(91)90171-Y) ranging from 18-68 Gy.

3.81.3.7.2 Default

- `"51.9"`

3.81.3.7.3 Examples

- `"51.9"`
- `"37.9"`

3.81.3.8 EUD_Alpha

3.81.3.8.1 Description The weighting factor α that controls the relative weighting of volume and dose in the generalized Equivalent Uniform Dose (gEUD) model. When $\alpha = 1$, the gEUD is equivalent to the mean; when $\alpha = 0$, the gEUD is equivalent to the geometric mean. Wu et al. (doi:10.1016/S0360-3016(01)02585-8) claim that for normal tissues, α can be related to the Lyman-Kutcher-Burman (LKB) model volume parameter 'n' via $\alpha = 1/n$. Sovik et al. (doi:10.1016/j.ejmp.2007.09.001) found that gEUD is not strongly impacted by error in α. Niemierko et al. ('A generalized concept of equivalent uniform dose. Med Phys 26:1100, 1999) generated maximum likelihood estimates for 'several tumors and normal structures' which ranged from -13.1 for local control of chordoma tumors to +17.7 for perforation of esophagus. Gay et al. (doi:10.1016/j.ejmp.2007.07.001) table 2 lists estimates based on the work of Emami (doi:10.1016/0360-3016(91)90171-Y) for normal tissues ranging from 1-31. Brenner et al. (doi:10.1016/0360-3016(93)90189-3) recommend -7.2 for breast cancer, -10 for melanoma, and -13 for squamous cell carcinomas. A 2017 presentation by Ontida Apinorasethkul claims the tumour range spans [-40:-1] and the organs at risk range spans [1:40]. AAPM TG report 166 also provides a listing of recommended values, suggesting -10 for PTV and GTV, +1 for parotid, 20 for spinal cord, and 8-16 for rectum, bladder, brainstem, chiasm, eye, and optic nerve. Burman (1991) and QUANTEC (2010) also provide estimates.

3.81.3.8.2 Default

- `"-13.0"`

3.81.3.8.3 Examples

- `"-40"`
- `"-13.0"`
- `"-10"`
- `"-7.2"`
- `"0.3"`
- `"1"`
- `"3"`
- `"4"`
- `"20"`

- "40"

3.81.3.9 Fenwick_C

3.81.3.9.1 Description This parameter describes the degree that superlinear doses are required to control large tumours. In other words, as tumour volume grows, a disproportionate amount of additional dose is required to maintain the same level of control. The Fenwick model is semi-empirical, so this number must be fitted or used from values reported in the literature. Fenwick et al. 2008 (doi:10.1016/j.clon.2008.12.011) provide values: 9.58 for local progression free survival at 30 months for NSCLC tumours and 5.00 for head-and-neck tumours.

3.81.3.9.2 Default

- "9.58"

3.81.3.9.3 Examples

- "9.58"
- "5.00"

3.81.3.10 Fenwick_M

3.81.3.10.1 Description This parameter describes the dose-response steepness in the Fenwick model. Fenwick et al. 2008 (doi:10.1016/j.clon.2008.12.011) provide values: 0.392 for local progression free survival at 30 months for NSCLC tumours and 0.280 for head-and-neck tumours.

3.81.3.10.2 Default

- "0.392"

3.81.3.10.3 Examples

- "0.392"
- "0.280"

3.81.3.11 Fenwick_Vref

3.81.3.11.1 Description This parameter is the volume (in DICOM units; usually mmˆ3) of a reference tumour (i.e., GTV; primary tumour and involved nodes) which the D_{50} are estimated using. In other words, this is a 'nominal' tumour volume. Fenwick et al. 2008 (doi:10.1016/j.clon.2008.12.011) recommend 148'410 mmˆ3 (i.e., a sphere of diameter 6.6 cm). However, an appropriate value depends on the nature of the tumour.

3.81.3.11.2 Default

- `"148410.0"`

3.81.3.11.3 Examples

- `"148410.0"`

3.81.3.12 UserComment

3.81.3.12.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. If left empty, the column will be omitted from the output.

3.81.3.12.2 Default

- `""`

3.81.3.12.3 Examples

- `""`
- `"Using XYZ"`
- `"Patient treatment plan C"`

3.82 ExportFITSImages

3.82.1 Description

This operation writes image arrays to FITS-formatted image files.

3.82.2 Notes

- Only pixel information and basic image positioning metadata are exported. In particular, contours and arbitrary metadata are **not** exported by this routine. (If a rendering of the image with contours drawn is needed, consult the PresentationImage operation.)

3.82.3 Parameters

- ImageSelection
- FilenameBase

3.82.3.1 ImageSelection

3.82.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.82.3.1.2 Default

- `"last"`

3.82.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.82.3.2 FilenameBase

3.82.3.2.1 Description The base filename that images will be written to. A sequentially-increasing number and file suffix are appended after the base

filename. Note that the file type is FITS.

3.82.3.2.2 Default

- `"/tmp/dcma_exportfitsimages"`

3.82.3.2.3 Examples

- `"../somedir/out"`
- `"/path/to/some/dir/file_prefix"`

3.83 ExportLineSamples

3.83.1 Description

This operation writes a line sample to a file.

3.83.2 Parameters

- LineSelection
- FilenameBase

3.83.2.1 LineSelection

3.83.2.1.1 Description Select one or more line samples. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth line sample (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last line sample. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the line sample composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all line sample that do not have the greatest number of sub-objects, not the least-numerous line sample (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.83.2.1.2 Default

- `"last"`

3.83.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.83.2.2 FilenameBase

3.83.2.2.1 Description The base filename that line samples will be written to. The file format is a 4-column text file that can be readily plotted. The columns are 'x dx f df' where dx (df) represents the uncertainty in x (f) if available. Metadata is included, but will be base64 encoded if any non-printable characters are detected. If no name is given, the default will be used. A '_', a sequentially-increasing number, and the'.dat' file suffix are appended after the base filename.

3.83.2.2.2 Default

- `"/tmp/dcma_exportlinesamples"`

3.83.2.2.3 Examples

- `"line_sample"`
- `"../somedir/data"`
- `"/path/to/some/line_sample_to_plot"`

3.84 ExportPointClouds

3.84.1 Description

This operation writes point clouds to file.

3.84.2 Parameters

- PointSelection
- FilenameBase

3.84.2.1 PointSelection

3.84.2.1.1 Description Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.84.2.1.2 Default

- `"last"`

3.84.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`

- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.84.2.2 FilenameBase

3.84.2.2.1 Description The base filename that line samples will be written to. The file format is 'XYZ' – a 3-column text file containing vector coordinates of the points. Metadata is excluded. A '_', a sequentially-increasing number, and the'.xyz' file suffix are appended after the base filename.

3.84.2.2.2 Default

- `"/tmp/dcma_exportpointclouds"`

3.84.2.2.3 Examples

- `"point_cloud"`
- `"../somedir/data"`
- `"/path/to/some/points"`

3.85 ExportSurfaceMeshes

3.85.1 Description

This operation writes one or more surface meshs to file in the 'Stanford' Polygon File format.

3.85.2 Notes

- Support for metadata in OBJ files is fully supported. Surface mesh metadata will be encoded in specially-marked comments and base64 encoded if non-printable characters are present. Metadata will be recovered when PLY files are loaded in DICOMautomaton. Note that other software may disregard these comments.

3.85.3 Parameters

- MeshSelection
- Filename
- Variant

3.85.3.1 MeshSelection

3.85.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.85.3.1.2 Default

- `"last"`

3.85.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.85.3.2 Filename

3.85.3.2.1 Description The filename (or full path name) to which the surface mesh data should be written. Existing files will not be overwritten. If an invalid or missing file extension is provided, one will automatically be added. If an empty filename is given, a unique name will be chosen automatically. If multiple meshes are selected, each will be written to a separate file; the name of each will be derived from the user-provided filename (or default) by appending a sequentially increasing counter between the file's stem name and extension. Files will be formatted in Stanford Polygon File ('PLY') format.

3.85.3.2.2 Default

- `""`

3.85.3.2.3 Examples

- `"surface_mesh.ply"`
- `"../somedir/mesh.ply"`
- `"/path/to/some/surface_mesh.ply"`

3.85.3.3 Variant

3.85.3.3.1 Description Controls whether files are written in the binary or ASCII PLY file format variants. Binary files will generally be smaller, and therefore faster to write, but may be less portable. ASCII format is better suited for archival purposes, and may be more widely supported. ASCII is generally recommended unless performance or storage will be problematic.

3.85.3.3.2 Default

- `"ascii"`

3.85.3.3.3 Supported Options

- `"ascii"`
- `"binary"`

3.86 ExportSurfaceMeshesOBJ

3.86.1 Description

This operation writes one or more surface meshes to file in Wavefront Object ('OBJ') format.

3.86.2 Notes

- Support for metadata in OBJ files is currently limited. Metadata will generally be lost.
- OBJ files can refer to MTL 'sidecar' files for information about materials and various properties. MTL files are not supported at this time.

3.86.3 Parameters

- MeshSelection
- Filename

3.86.3.1 MeshSelection

3.86.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.86.3.1.2 Default

- `"last"`

3.86.3.1.3 Examples

- `"last"`

- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.86.3.2 Filename

3.86.3.2.1 Description The filename (or full path name) to which the surface mesh data should be written. Existing files will not be overwritten. If an invalid or missing file extension is provided, one will automatically be added. If an empty filename is given, a unique name will be chosen automatically. If multiple meshes are selected, each will be written to a separate file; the name of each will be derived from the user-provided filename (or default) by appending a sequentially increasing counter between the file's stem name and extension. Files will be formatted in ASCII Wavefront Object ('OBJ') format.

3.86.3.2.2 Default

- `""`

3.86.3.2.3 Examples

- `"surface_mesh.obj"`
- `"../somedir/mesh.obj"`
- `"/path/to/some/surface_mesh.obj"`

3.87 ExportSurfaceMeshesOFF

3.87.1 Description

This operation writes one or more surface meshes to file in Object File Format ('OFF').

3.87.2 Notes

- Support for metadata in OFF files is currently limited. Metadata will generally be lost.

- OFF files can contain many different types of geometry, and some software may not support the specific subset used by DICOMautomaton. For example, vertex normals may not be supported, and their presence can cause some OFF file loaders to reject valid OFF files. For the best portability, consider more common formats like PLY or OBJ.

3.87.3 Parameters

- MeshSelection
- Filename

3.87.3.1 MeshSelection

3.87.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.87.3.1.2 Default

- `"last"`

3.87.3.1.3 Examples

- `"last"`
- `"first"`

- "all"
- "none"
- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.87.3.2 Filename

3.87.3.2.1 Description The filename (or full path name) to which the surface mesh data should be written. Existing files will not be overwritten. If an invalid or missing file extension is provided, one will automatically be added. If an empty filename is given, a unique name will be chosen automatically. If multiple meshes are selected, each will be written to a separate file; the name of each will be derived from the user-provided filename (or default) by appending a sequentially increasing counter between the file's stem name and extension. Files will be formatted in Object File Format ('OFF').

3.87.3.2.2 Default

- ""

3.87.3.2.3 Examples

- "surface_mesh.off"
- "../somedir/mesh.off"
- "/path/to/some/surface_mesh.off"

3.88 ExportSurfaceMeshesPLY

3.88.1 Description

This operation writes one or more surface meshs to file in the 'Stanford' Polygon File format.

3.88.2 Notes

- Support for metadata in OBJ files is fully supported. Surface mesh metadata will be encoded in specially-marked comments and base64 encoded if non-printable characters are present. Metadata will be recovered when PLY files are loaded in DICOMautomaton. Note that other software may disregard these comments.

3.88.3 Parameters

- MeshSelection
- Filename
- Variant

3.88.3.1 MeshSelection

3.88.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.88.3.1.2 Default

- `"last"`

3.88.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`

- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.88.3.2 Filename

3.88.3.2.1 Description The filename (or full path name) to which the surface mesh data should be written. Existing files will not be overwritten. If an invalid or missing file extension is provided, one will automatically be added. If an empty filename is given, a unique name will be chosen automatically. If multiple meshes are selected, each will be written to a separate file; the name of each will be derived from the user-provided filename (or default) by appending a sequentially increasing counter between the file's stem name and extension. Files will be formatted in Stanford Polygon File ('PLY') format.

3.88.3.2.2 Default

- ""

3.88.3.2.3 Examples

- "surface_mesh.ply"
- "../somedir/mesh.ply"
- "/path/to/some/surface_mesh.ply"

3.88.3.3 Variant

3.88.3.3.1 Description Controls whether files are written in the binary or ASCII PLY file format variants. Binary files will generally be smaller, and therefore faster to write, but may be less portable. ASCII format is better suited for archival purposes, and may be more widely supported. ASCII is generally recommended unless performance or storage will be problematic.

3.88.3.3.2 Default

- "ascii"

3.88.3.3.3 Supported Options

- "ascii"
- "binary"

3.89 ExportSurfaceMeshesSTL

3.89.1 Description

This operation writes one or more surface meshes to file in the (3D Systems) Stereolithography format.

3.89.2 Notes

- Support for metadata in STL files is currently limited. Metadata will generally be lost.
- The STL format is generally meant to be sent to hardware with limited processing power or memory, and is pre-processed so that individual faces can be easily streamed. This pre-processing destroys information about the mesh, for example face adjacency. This information can be hard or impossible to fully recover. If you need to later process, or re-process a surface mesh, avoid the STL file format if possible. Alternatives supported by DICOMautomaton include PLY, OBJ, and OFF formats.

3.89.3 Parameters

- MeshSelection
- Filename
- Variant

3.89.3.1 MeshSelection

3.89.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that

'!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.89.3.1.2 Default

- `"last"`

3.89.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.89.3.2 Filename

3.89.3.2.1 Description The filename (or full path name) to which the surface mesh data should be written. Existing files will not be overwritten. If an invalid or missing file extension is provided, one will automatically be added. If an empty filename is given, a unique name will be chosen automatically. If multiple meshes are selected, each will be written to a separate file; the name of each will be derived from the user-provided filename (or default) by appending a sequentially increasing counter between the file's stem name and extension. Files will be formatted in Stereolithography ('STL') format.

3.89.3.2.2 Default

- `""`

3.89.3.2.3 Examples

- `"surface_mesh.stl"`
- `"../somedir/mesh.stl"`
- `"/path/to/some/surface_mesh.stl"`

3.89.3.3 Variant

3.89.3.3.1 Description Controls whether files are written in the binary or ASCII STL file format variants. Binary files will generally be smaller, and therefore faster to write, but may be less portable. ASCII format is better suited for archival purposes, and may be more widely supported. ASCII is generally recommended unless performance or storage will be problematic.

3.89.3.3.2 Default

- `"ascii"`

3.89.3.3.3 Supported Options

- `"ascii"`
- `"binary"`

3.90 ExportWarps

3.90.1 Description

This operation exports a transform object (e.g., affine matrix, vector deformation field) to file.

3.90.2 Parameters

- TransformSelection
- Filename

3.90.2.1 TransformSelection

3.90.2.1.1 Description The transformation that will be exported. Select one or more transform objects (aka 'warp' objects). Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth transformation (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last transformation. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the transformation composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that

'!numerous' means all transformation that do not have the greatest number of sub-objects, not the least-numerous transformation (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.90.2.1.2 Default

- `"last"`

3.90.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`

3.90.2.2 Filename

3.90.2.2.1 Description The filename (or full path name) to which the transformation should be written. Existing files will be overwritten. The file format is a 4x4 Affine matrix. If no name is given, a unique name will be chosen automatically.

3.90.2.2.2 Default

- `""`

3.90.2.2.3 Examples

- `"transformation.trans"`
- `"trans.txt"`
- `"/path/to/some/trans.txt"`

3.91 ExtractAlphaBeta

3.91.1 Description

This operation compares two images arrays: either a biologically-equivalent dose ($BED_{\alpha/\beta}$) transformed array or an equivalent dose in d dose per fraction

(EQD_x) array and a 'reference' untransformed array. The α/β used for each voxel are extracted by comparing corresponding voxels. Each voxel is overwritten with the value of α/β needed to accomplish the given transform. This routine is best used to inspect a given transformation (e.g., for QA purposes).

3.91.2 Notes

- Images are overwritten, but ReferenceImages are not. Multiple Images may be specified, but only one ReferenceImages may be specified.
- The reference image array must be rectilinear. (This is a requirement specific to this implementation, a less restrictive implementation could overcome the issue.)
- For the fastest and most accurate results, test and reference image arrays should spatially align. However, alignment is **not** necessary. If test and reference image arrays are aligned, image adjacency can be precomputed and the analysis will be faster. If not, image adjacency must be evaluated for each image slice. If this also fails, it will be evaluated for every voxel.
- This operation will make use of interpolation if corresponding voxels do not exactly overlap.

3.91.3 Parameters

- TransformedImageSelection
- ReferenceImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Model
- Channel
- TestImgLowerThreshold
- TestImgUpperThreshold
- NumberOfFractions
- NominalDosePerFraction

3.91.3.1 TransformedImageSelection

3.91.3.1.1 Description The transformed image array where voxel intensities represent BED or EQDd. Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based

indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.91.3.1.2 Default

- `"first"`

3.91.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.91.3.2 ReferenceImageSelection

3.91.3.2.1 Description The un-transformed image array where voxel intensities represent (non-BED) dose. Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based

indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.91.3.2.2 Default

- `"last"`

3.91.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.91.3.3 NormalizedROILabelRegex

3.91.3.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.91.3.3.2 Default

- `".*"`

3.91.3.3.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.91.3.4 ROILabelRegex

3.91.3.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.91.3.4.2 Default

- `".*"`

3.91.3.4.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.91.3.5 Model

3.91.3.5.1 Description The model of BED or EQDx transformation to assume. Currently, only 'eqdx-lq-simple' is available. The 'eqdx-lq-simple' model does not take into account elapsed time or any cell repopulation effects.

3.91.3.5.2 Default

- `"eqdx-lq-simple"`

3.91.3.5.3 Supported Options

- `"eqdx-lq-simple"`

3.91.3.6 Channel

3.91.3.6.1 Description The channel to compare (zero-based). Setting to -1 will compare each channel separately. Note that both test images and reference images must share this specifier.

3.91.3.6.2 Default

- `"0"`

3.91.3.6.3 Examples

- `"-1"`
- `"0"`
- `"1"`
- `"2"`

3.91.3.7 TestImgLowerThreshold

3.91.3.7.1 Description Pixel lower threshold for the test images. Only voxels with values above this threshold (inclusive) will be altered.

3.91.3.7.2 Default

- `"-inf"`

3.91.3.7.3 Examples

- `"-inf"`
- `"0.0"`
- `"200"`

3.91.3.8 TestImgUpperThreshold

3.91.3.8.1 Description Pixel upper threshold for the test images. Only voxels with values below this threshold (inclusive) will be altered.

3.91.3.8.2 Default

- `"inf"`

3.91.3.8.3 Examples

- `"inf"`
- `"1.23"`
- `"1000"`

3.91.3.9 NumberOfFractions

3.91.3.9.1 Description Number of fractions assumed in the BED or EQDd transformation.

3.91.3.9.2 Default

- `"35"`

3.91.3.9.3 Examples

- `"1"`
- `"5"`
- `"35"`

3.91.3.10 NominalDosePerFraction

3.91.3.10.1 Description The nominal dose per fraction (in DICOM units; Gy) assumed by an EQDx transformation. This parameter is the 'x' in 'EQDx'; for EQD2 transformations, this parameter must be 2 Gy.

3.91.3.10.2 Default

- `"2.0"`

3.91.3.10.3 Examples

- `"1.8"`
- `"2.0"`
- `"8.0"`

3.92 ExtractImageHistograms

3.92.1 Description

This operation extracts histograms (e.g., dose-volume – DVH, or pixel intensity-volume) for the selected image(s) and ROI(s). Results are stored as line samples for later analysis or export.

3.92.2 Notes

- This routine generates differential histograms with unscaled abscissae and ordinate axes. It also generates cumulative histograms with unscaled abscissae and *both* unscaled and peak-normalized-to-one ordinates. Unscaled abscissa are reported in DICOM units (typically HU or Gy), unscaled ordinates are reported in volumetric DICOM units (mm^3), and normalized ordinates are reported as a fraction of the given ROI's total volume.
- Non-finite voxels are excluded from analysis and do not contribute to the volume. If exact volume is required, ensure all voxels are finite prior to invoking this routine.
- This routine can handle contour partitions where the physical layout (i.e., storage order) differs from the logical layout. See the 'grouping' options for available configuration.
- This routine will correctly handle non-overlapping voxels with varying volumes (i.e., rectilinear image arrays). It will *not* correctly handle overlapping voxels (i.e., each overlapping voxel will be counted without regard for overlap). If necessary, resample image arrays to be rectilinear.

3.92.3 Parameters

- ImageSelection
- Channel
- ROILabelRegex
- NormalizedROILabelRegex
- ContourOverlap
- Inclusivity
- Grouping
- GroupLabel
- Lower
- Upper
- dDose
- UserComment

3.92.3.1 ImageSelection

3.92.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.92.3.1.2 Default

- `"last"`

3.92.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.92.3.2 Channel

3.92.3.2.1 Description The image channel to use. Zero-based. Use '-1' to operate on all available channels.

3.92.3.2.2 Default

- `"-1"`

3.92.3.2.3 Examples

- `"-1"`
- `"0"`
- `"1"`
- `"2"`

3.92.3.3 ROILabelRegex

3.92.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.92.3.3.2 Default

- `".*"`

3.92.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.92.3.4 NormalizedROILabelRegex

3.92.3.4.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single)

regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.92.3.4.2 Default

- `".*"`

3.92.3.4.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.92.3.5 ContourOverlap

3.92.3.5.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.92.3.5.2 Default

- `"ignore"`

3.92.3.5.3 Supported Options

- `"ignore"`
- `"honour_opposite_orientations"`
- `"overlapping_contours_cancel"`
- `"honour_opps"`
- `"overlap_cancel"`

3.92.3.6 Inclusivity

3.92.3.6.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.92.3.6.2 Default

- `"center"`

3.92.3.6.3 Supported Options

- `"center"`
- `"centre"`
- `"planar_corner_inclusive"`
- `"planar_inc"`
- `"planar_corner_exclusive"`
- `"planar_exc"`

3.92.3.7 Grouping

3.92.3.7.1 Description This routine partitions individual contours using their ROI labels. This parameter controls whether contours with different names should be treated as though they belong to distinct logical groups ('separate') or whether *all* contours should be treated as though they belong to a single logical group ('combined'). The 'separate' option works best for exploratory analysis, extracting histograms for many OARs at once, or when you know the 'physical' grouping of contours by label reflects a consistent logical grouping. The 'combined' option works best when the physical and logical groupings are inconsistent. For example, when you need a combined histograms from multiple contours or organs, or when similar structures should be combined (e.g., spinal cord + canal; or distinct left + right lateral organs that should be paired, e.g.. 'combined parotids'). Note that when the 'combined' option is used, the 'GroupLabel' parameter *must* also be provided.

3.92.3.7.2 Default

- `"separate"`

3.92.3.7.3 Supported Options

- `"separate"`
- `"grouped"`

3.92.3.8 GroupLabel

3.92.3.8.1 Description If the 'Grouping' parameter is set to 'combined', the value of the 'GroupLabel' parameter will be used in lieu of any consitituent ROILabel. Note that this parameter *must* be provided when the 'Grouping' parameter is set to'combined'.

3.92.3.8.2 Default

- `""`

3.92.3.8.3 Examples

- `"combination"`
- `"multiple_rois"`
- `"logical_oar"`
- `"both_oars"`

3.92.3.9 Lower

3.92.3.9.1 Description Disregard all voxel values lower than this value. This parameter can be used to filter out spurious values. All voxels with infinite or NaN intensities are excluded regardless of this parameter. Note that disregarded values will not contribute any volume.

3.92.3.9.2 Default

- `"-inf"`

3.92.3.9.3 Examples

- `"-inf"`
- `"-100.0"`
- `"0.0"`
- `"1.2"`
- `"5.0E23"`

3.92.3.10 Upper

3.92.3.10.1 Description Disregard all voxel values greater than this value. This parameter can be used to filter out spurious values. All voxels with infinite or NaN intensities are excluded regardless of this parameter. Note that disregarded values will not contribute any volume.

3.92.3.10.2 Default

- `"inf"`

3.92.3.10.3 Examples

- "-100.0"
- "0.0"
- "1.2"
- "5.0E23"
- "inf"

3.92.3.11 dDose

3.92.3.11.1 Description The (fixed) bin width, in units of dose (DICOM units; nominally Gy). Note that this is the *maximum* bin width, in practice bins may be smaller to account for slop (i.e., excess caused by the extrema being separated by a non-integer number of bins of width *dDose*).

3.92.3.11.2 Default

- "0.1"

3.92.3.11.3 Examples

- "0.0001"
- "0.001"
- "0.01"
- "5.0"
- "10"
- "50"

3.92.3.12 UserComment

3.92.3.12.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data. If left empty, the column will be omitted from the output.

3.92.3.12.2 Default

- ""

3.92.3.12.3 Examples

- "Using XYZ"
- "Patient treatment plan C"

3.93 ExtractPointsWarp

3.93.1 Description

This operation uses two point clouds (one 'moving' and the other 'stationary' or 'reference') to find a transformation ('warp') that will map the moving point set to the stationary point set. The resulting transformation encapsulates a 'registration' between the two point sets – however the transformation is generic and can be later be used to move (i.e., 'warp', 'deform') other objects, including the 'moving' point set.

3.93.2 Notes

- The 'moving' point cloud is *not* warped by this operation – this operation merely identifies a suitable transformation. Separation of the identification and application of a warp allows the warp to more easily re-used and applied to multiple objects.
- The output of this operation is a transformation that can later be applied, in principle, to point clouds, surface meshes, images, arbitrary vector fields, and any other objects in R^3.
- There are multiple algorithms implemented. Some do *not* provide bijective mappings, meaning that swapping the inputs will result in an altogether different registration (even after inverting it).

3.93.3 Parameters

- MovingPointSelection
- ReferencePointSelection
- Method
- TPSLambda
- TPSKernelDimension
- TPSSolver
- TPSRPMLambdaStart
- TPSRPMZetaStart
- TPSRPMDoubleSidedOutliers
- TPSRPMKernelDimension
- TPSRPMTStart
- TPSRPMTEnd
- TPSRPMTStep
- TPSRPMStepsPerT
- TPSRPMSinkhornMaxSteps
- TPSRPMSinkhornTolerance
- TPSRPMSeedWithCentroidShift
- TPSRPMSolver
- TPSRPMHardConstraints
- TPSRPMPermitMovingOutliers

- TPSRPMPermitStationaryOutliers
- MaxIterations
- RelativeTolerance

3.93.3.1 MovingPointSelection

3.93.3.1.1 Description The point cloud that will serve as input to the warp function. Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.93.3.1.2 Default

- `"last"`

3.93.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`

- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.93.3.2 ReferencePointSelection

3.93.3.2.1 Description The stationary point cloud to use as a reference for the moving point cloud. Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified. Note that this point cloud is not modified.

3.93.3.2.2 Default

- `"last"`

3.93.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`

- `"numerous"`

3.93.3.3 Method

3.93.3.3.1 Description The alignment algorithm to use. The following alignment options are available: 'centroid', 'PCA', 'exhaustive_icp', 'TPS', and 'TPS-RPM'. The 'centroid' option finds a rotationless translation the aligns the centroid (i.e., the centre of mass if every point has the same 'mass') of the moving point cloud with that of the stationary point cloud. It is susceptible to noise and outliers, and can only be reliably used when the point cloud has complete rotational symmetry (i.e., a sphere). On the other hand, 'centroid' alignment should never fail, can handle a large number of points, and can be used in cases of 2D and 1D degeneracy. centroid alignment is frequently used as a pre-processing step for more advanced algorithms. The 'PCA' option finds an Affine transformation by performing centroid alignment, performing principle component analysis (PCA) separately on the reference and moving point clouds, computing third-order point distribution moments along each principle axis to establish a consistent orientation, and then rotates the moving point cloud so the principle axes of the stationary and moving point clouds coincide. The 'PCA' method may be suitable when: (1) both clouds are not contaminated with extra noise points (but some Gaussian noise in the form of point 'jitter' should be tolerated) and (2) the clouds are not perfectly spherical (i.e., so they have valid principle components). However, note that the 'PCA' method is susceptible to outliers and can not scale a point cloud. The 'PCA' method will generally fail when the distribution of points shifts across the centroid (i.e., comparing reference and moving point clouds) since the orientation of the components will be inverted, however 2D degeneracy is handled in a 3D-consistent way, and 1D degeneracy is handled in a 1D-consistent way (i.e, the components orthogonal to the common line will be completely ambiguous, so spurious rotations will result). The 'exhaustive_icp' option finds an Affine transformation by first performing PCA-based alignment and then iteratively alternating between (1) estimating point-point correspondence and (1) solving for a least-squares optimal transformation given this correspondence estimate. 'ICP' stands for 'iterative closest point.' Each iteration uses the previous transformation *only* to estimate correspondence; a least-squares optimal linear transform is estimated afresh each iteration. The 'exhaustive_icp' method is most suitable when both point clouds consist of approximately 50k points or less. Beyond this, ICP will still work but runtime scales badly. ICP is susceptible to outliers and will not scale a point cloud. It can be used for 2D and 1D degenerate problems, but is not guaranteed to find the 'correct' orientation of degenerate or symmetrical point clouds. The 'TPS' or Thin-Plate Spline algorithm provides non-rigid (i.e., 'deformable') registration between corresponding point sets. The moving and stationary point sets must have the same number of points, and the n^{th} moving point is taken to correspond to the n^{th} stationary point. The 'TPS' method does not scale well due in part to inversion of a large (NxN) matrix and is therefore most suitable when both point

clouds consist of approximately 10-20k points or less. Beyond this, expect slow calculations. The TPS method is not robust to outliers, however a regularization parameter can be used to control the smoothness of the warp. (Setting to zero will cause the warp function to exactly interpolate every pair, except due to floating point inaccuracies.) Also note that the TPS method can only, in general, be used for interpolation. Extrapolation beyond the points clouds will almost certainly result in wildly inconsistent and unstable transformations. Consult Bookstein 1989 (doi:10.1109/34.24792) for an overview. The 'TPS-RPM' or Thin-Plate Spline Robust Point-Matching algorithm provides non-rigid (i.e., 'deformable') registration. It combines a soft-assign technique, deterministic annealing, and thin-plate splines to iteratively solve for correspondence and spatial warp. The 'TPS-RPM' method is (somewhat) robust to outliers in both moving and stationary point sets, but it suffers from numerical instabilities when one or more inputs are degenerate or symmetric in such a way that many potential solutions have the same least-square cost. The 'TPS-RPM' method does not scale well due in part to inversion of a large (NxM) matrix and is therefore most suitable when both point clouds consist of approximately 1-5k points or less. Beyond this, expect slow calculations. Also note that the underlying TPS method can only, in general, be used for interpolation. Extrapolation beyond the extent of the corresponding parts of the points clouds will almost certainly result in wildly inconsistent and unstable transformations. Consult Chui and Rangarajan 2000 (original algorithm; doi:10.1109/CVPR.2000.854733) and Yang 2011 (clarification and more robust solution; doi:10.1016/j.patrec.2011.01.015) for more details.

3.93.3.3.2 Default

- `"centroid"`

3.93.3.3.3 Supported Options

- `"centroid"`
- `"pca"`
- `"exhaustive_icp"`
- `"tps"`
- `"tps_rpm"`

3.93.3.4 TPSLambda

3.93.3.4.1 Description Regularization parameter for the TPS method. Controls the smoothness of the fitted thin plate spline function. Setting to zero will ensure that all points are interpolated exactly (barring numerical imprecision). Setting higher will allow the spline to 'relax' and smooth out. The specific value to use is heavily dependent on the problem domain and the amount of noise and outliers in the data. It relates to the spacing between points. Note that this parameter is used with the TPS method, but *not* in the TPS-RPM method.

3.93.3.4.2 Default

- `"0.0"`

3.93.3.4.3 Examples

- `"1E-4"`
- `"0.1"`
- `"10.0"`

3.93.3.5 TPSKernelDimension

3.93.3.5.1 Description Dimensionality of the spline function kernel. The kernel dimensionality *should* match the dimensionality of the points (i.e., 3), but doesn't need to. 2 seems to work best, even with points in 3D. Note that this parameter may affect how the transformation extrapolates.

3.93.3.5.2 Default

- `"2"`

3.93.3.5.3 Examples

- `"2"`
- `"3"`

3.93.3.6 TPSSolver

3.93.3.6.1 Description The method used to solve the system of linear equtions that defines the thin plate spline solution. The pseudoinverse will likely be able to provide a solution when the system is degenerate, but it might not be reasonable or even sensible. The LDLT method scales better.

3.93.3.6.2 Default

- `"LDLT"`

3.93.3.6.3 Supported Options

- `"LDLT"`
- `"PseudoInverse"`

3.93.3.7 TPSRPMLambdaStart

3.93.3.7.1 Description Regularization parameter for the TPS-RPM method. Controls the smoothness of the fitted thin plate spline function. Setting to zero will ensure that all points are interpolated exactly (barring numerical imprecision). Setting higher will allow the spline to 'relax' and smooth out. The specific value to use is heavily dependent on the problem domain and the amount of noise and outliers in the data. It relates to the spacing between points. It follows the same annealing schedule as the system temperature does. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.7.2 Default

- `"0.0"`

3.93.3.7.3 Examples

- `"0.0"`
- `"1E-4"`
- `"0.1"`
- `"10.0"`

3.93.3.8 TPSRPMZetaStart

3.93.3.8.1 Description Regularization parameter for the TPS-RPM method. Controls the likelihood of points being treated as outliers. Higher values will bias points towards *not* being considered outliers. The specific value to use is heavily dependent on the problem domain and the amount of noise and outliers in the data. It relates to the spacing between points. It follows the same annealing schedule as the system temperature does. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.8.2 Default

- `"0.0"`

3.93.3.8.3 Examples

- `"0.0"`
- `"1E-4"`
- `"0.1"`
- `"10.0"`

3.93.3.9 TPSRPMDoubleSidedOutliers

3.93.3.9.1 Description Controls whether the extensions for 'double sided outlier handling' as described by Yang et al. (2011; doi:10.1016/j.patrec.2011.01.015) are used. These extensions can improve resilience to outliers, especially in the moving set. Yang et al. also mention that the inclusion of an extra entropy term in the cost function can help reduce jitter during the annealing process, which may result in fewer folds or twists for narrow point clouds. However, the resulting algorithm is overall less numerically stable and has a strong dependence on the kernel dimension. Enabling this parameter adjusts the interpretation of the lambda regularization parameter, so some fine-tuning may be required. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.9.2 Default

- `"false"`

3.93.3.9.3 Examples

- `"true"`
- `"false"`

3.93.3.10 TPSRPMKernelDimension

3.93.3.10.1 Description Dimensionality of the spline function kernel. The kernel dimensionality *should* match the dimensionality of the points (i.e., 3), but doesn't need to. 2 seems to work best, even with points in 3D. Note that this parameter may affect how the transformation extrapolates. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.10.2 Default

- `"2"`

3.93.3.10.3 Supported Options

- `"2"`
- `"3"`

3.93.3.11 TPSRPMTStart

3.93.3.11.1 Description The deterministic annealing starting temperature. This parameter is a scaling factor that modifies the temperature determined via an automatic method. Larger numbers grant the system more freedom to find large-scale deformation; small values *limit* the freedom to find large-scale deformations. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.11.2 Default

- `"1.05"`

3.93.3.11.3 Examples

- `"1.5"`
- `"1.05"`
- `"0.8"`
- `"0.5"`

3.93.3.12 TPSRPMTEnd

3.93.3.12.1 Description The deterministic annealing ending temperature. Higher numbers will result in a coarser, but faster registration. This parameter is a scaling factor that modifies the temperature determined via an automatic method. Larger numbers limit the freedom of the system to find fine-detail deformations; small values may result in overfitting and folding deformations. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.12.2 Default

- `"0.01"`

3.93.3.12.3 Examples

- `"1.0"`
- `"0.1"`
- `"0.01"`

3.93.3.13 TPSRPMTStep

3.93.3.13.1 Description The deterministic annealing ending temperature. Higher numbers will result in slower annealing. This parameter is a multiplicative factor, so if set to 0.95 temperature adjustments will be $T' = 0.95T$. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.13.2 Default

- `"0.93"`

3.93.3.13.3 Examples

- `"0.99"`
- `"0.93"`
- `"0.9"`

3.93.3.14 TPSRPMStepsPerT

3.93.3.14.1 Description Deterministic annealing parameter controlling the number of correspondence-transformation update iterations performed at each temperature. Lower numbers will result in faster, but possibly less accurate registrations. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.14.2 Default

- `"5"`

3.93.3.14.3 Examples

- `"1"`
- `"5"`
- `"10"`

3.93.3.15 TPSRPMSinkhornMaxSteps

3.93.3.15.1 Description Parameter controlling the number of iterations performed during the Sinkhorn softassign correspondence estimation procedure. Note that this is the worst-case number of iterations since the Sinkhorn procedure completes when tolerance is reached. Setting this number to the maximum number of iterations acceptable given your speed requirements should result in satisfactory results. Note that use of forced correspondence *may* require a higher number of steps. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.15.2 Default

- `"5000"`

3.93.3.15.3 Examples

- `"500"`
- `"5000"`
- `"50000"`
- `"500000"`

3.93.3.16 TPSRPMSinkhornTolerance

3.93.3.16.1 Description Parameter controlling the permissable deviation from the ideal softassign correspondence normalization conditions (i.e., that each row and each column sum to one). If tolerance is reached then the Sinkhorn procedure is completed early. However, if the maximum number of iterations is

reached and the tolerance has not been achieved then the algorithm terminates due to failure. If registration quality is flexible, setting a higher number can significantly speed up the computation. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.16.2 Default

- `"0.01"`

3.93.3.16.3 Examples

- `"1E-4"`
- `"0.001"`
- `"0.01"`

3.93.3.17 TPSRPMSeedWithCentroidShift

3.93.3.17.1 Description Controls whether a centroid-based registration is used to seed the registration. Typically this is not needed, since high temperatures give the system enough freedom to find large-scale deformations (include centroid alignment). However, if the initial alignment is intentional, and point cloud centroids do not align, then seeding the registration will be detrimental. Seeding might be useful if the starting temperature is set low (which will limit large-scale deformations like centroid alignment). Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.17.2 Default

- `"false"`

3.93.3.17.3 Examples

- `"true"`
- `"false"`

3.93.3.18 TPSRPMSolver

3.93.3.18.1 Description The method used to solve the system of linear equtions that defines the thin plate spline solution. The pseudoinverse will likely be able to provide a solution when the system is degenerate, but it might not be reasonable or even sensible. The LDLT method scales better. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.18.2 Default

- `"LDLT"`

3.93.3.18.3 Supported Options

- `"LDLT"`
- `"PseudoInverse"`

3.93.3.19 TPSRPMHardConstraints

3.93.3.19.1 Description Forced correspondence between pairs of points (one in the moving set, one in the stationary set) specified as comma-separated pairs of indices into the moving and stationary point sets. Indices are zero-based. Forced correspondences are taken to be exclusive, meaning that no other points will correspond with either points. Forced correspondence also begets outlier rejection, so ensure the points are not tainted by noise or are outliers. Note that points can be forced to be treated as outliers by indicating a non-existent index in the opposite set, such as -1. Use of forced correspondence may cause the Sinkhorn method to converge slowly or possibly fail to converge at all. Increasing the number of Sinkhorn iterations may be required. Marking points as outliers has ramifications within the algorithm that can lead to numerical instabilities (especially in the moving point set). If possible, it is best to remove known outlier points *prior* to attempting registration. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.19.2 Default

- `""`

3.93.3.19.3 Examples

- `"0,10"`
- `"23,45, 24,46, 0,100, -1,50, 20,-1"`

3.93.3.20 TPSRPMPermitMovingOutliers

3.93.3.20.1 Description If enabled, this option permits the TPS-RPM algorithm to automatically detect and eschew outliers in the moving point set. A major strength of the TPS-RPM algorithm is that it can handle outliers, however there are legitimate cases where outliers are known *not* to be present, but the point-to-point correspondence is *not* known. Note that outlier detection cannot be used when one or more points are forced to be outliers. Similar to forced correspondence (i.e., hard constraints), disabling outlier detection can modify the Sinkhorn algorithm convergence. Additionally, Sinkhorn normalization is likely to fail when outliers in the larger point cloud are disallowed. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.20.2 Default

- `"true"`

3.93.3.20.3 Examples

- `"true"`
- `"false"`

3.93.3.21 TPSRPMPermitStationaryOutliers

3.93.3.21.1 Description If enabled, this option permits the TPS-RPM algorithm to automatically detect and eschew outliers in the stationary point set. A major strength of the TPS-RPM algorithm is that it can handle outliers, however there are legitimate cases where outliers are known *not* to be present, but the point-to-point correspondence is *not* known. Note that outlier detection cannot be used when one or more points are forced to be outliers. Similar to forced correspondence (i.e., hard constraints), disabling outlier detection can modify the Sinkhorn algorithm convergence. Additionally, Sinkhorn normalization is likely to fail when outliers in the larger point cloud are disallowed. Note that this parameter is used with the TPS-RPM method, but *not* in the TPS method.

3.93.3.21.2 Default

- `"true"`

3.93.3.21.3 Examples

- `"true"`
- `"false"`

3.93.3.22 MaxIterations

3.93.3.22.1 Description If the method is iterative, only permit this many iterations to occur. Note that this parameter will not have any effect on non-iterative methods.

3.93.3.22.2 Default

- `"100"`

3.93.3.22.3 Examples

- `"5"`
- `"20"`
- `"100"`
- `"1000"`

3.93.3.23 RelativeTolerance

3.93.3.23.1 Description If the method is iterative, terminate the loop when the cost function changes between successive iterations by this amount or less. The magnitude of the cost function will generally depend on the number of points (in both point clouds), the scale (i.e., 'width') of the point clouds, the amount of noise and outlier points, and any method-specific parameters that impact the cost function (if applicable); use of this tolerance parameter may be impacted by these characteristics. Verifying that a given tolerance is of appropriate magnitude is recommended. Relative tolerance checks can be disabled by setting to non-finite or negative value. Note that this parameter will only have effect on iterative methods that are not controlled by, e.g., an annealing schedule.

3.93.3.23.2 Default

- `"nan"`

3.93.3.23.3 Examples

- `"-1"`
- `"1E-2"`
- `"1E-3"`
- `"1E-5"`

3.94 ExtractRadiomicFeatures

3.94.1 Description

This operation extracts radiomic features from the selected images. Features are implemented as per specification in the Image Biomarker Standardisation Initiative (IBSI) or pyradiomics documentation if the IBSI specification is unclear or ambiguous.

3.94.2 Notes

- This routine is meant to be processed by an external analysis.
- If this routine is slow, simplifying ROI contours may help speed surface-mesh-based feature extraction. Often removing the highest-frequency components of the contour will help, such as edges that conform tightly to individual voxels.

3.94.3 Parameters

- UserComment
- FeaturesFileName
- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex

3.94.3.1 UserComment

3.94.3.1.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data.

3.94.3.1.2 Default

- ""

3.94.3.1.3 Examples

- ""
- "Using XYZ"
- "Patient treatment plan C"

3.94.3.2 FeaturesFileName

3.94.3.2.1 Description Features will be appended to this file. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.94.3.2.2 Default

- ""

3.94.3.2.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.94.3.3 ImageSelection

3.94.3.3.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.94.3.3.2 Default

- `"last"`

3.94.3.3.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.94.3.4 NormalizedROILabelRegex

3.94.3.4.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful

for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.94.3.4.2 Default

- `".*"`

3.94.3.4.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.94.3.5 ROILabelRegex

3.94.3.5.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.94.3.5.2 Default

- `".*"`

3.94.3.5.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.95 FVPicketFence

3.95.1 Description

This operation performs a picket fence QA test using an RTIMAGE file.

3.95.2 Notes

- This is a 'simplified' version of the full picket fence analysis program that uses defaults that are expected to be reasonable across a wide range of scenarios.

3.95.3 Parameters

- ROILabel
- ImageSelection
- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Channel
- Replacement
- Replace
- NeighbourCount
- AgreementCount
- MaxDistance
- ImageSelection
- DICOMMargin
- RTIMAGE
- ImageSelection
- RowsL
- RowsH
- ColumnsL
- ColumnsH
- DICOMMargin
- ImageSelection
- MLCModel
- MLCROILabel
- JunctionROILabel
- PeakROILabel
- MinimumJunctionSeparation
- ThresholdDistance
- LeafGapsFileName
- ResultsSummaryFileName
- UserComment
- InteractivePlots
- ScaleFactor
- ImageFileName

- ColourMapRegex
- WindowLow
- WindowHigh

3.95.3.1 ROILabel

3.95.3.1.1 Description A label to attach to the ROI contours.

3.95.3.1.2 Default

- `"entire_image"`

3.95.3.1.3 Examples

- `"everything"`
- `"whole_images"`
- `"unspecified"`

3.95.3.2 ImageSelection

3.95.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.95.3.2.2 Default

- `"last"`

3.95.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.95.3.3 ImageSelection

3.95.3.3.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.95.3.3.2 Default

- `"last"`

3.95.3.3.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.95.3.4 NormalizedROILabelRegex

3.95.3.4.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.95.3.4.2 Default

- `".*"`

3.95.3.4.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.95.3.5 ROILabelRegex

3.95.3.5.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.95.3.5.2 Default

- `"entire_image"`

3.95.3.5.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.95.3.6 Channel

3.95.3.6.1 Description The channel to operated on (zero-based). Negative values will cause all channels to be operated on.

3.95.3.6.2 Default

- `"0"`

3.95.3.6.3 Examples

- `"-1"`
- `"0"`
- `"1"`

3.95.3.7 Replacement

3.95.3.7.1 Description Controls how replacements are generated. 'Mean' and 'median' replacement strategies replace the voxel value with the mean and median, respectively, from the surrounding neighbourhood. 'Conservative' refers to the so-called conservative filter that suppresses isolated peaks; for every voxel

considered, the voxel intensity is clamped to the local neighbourhood's extrema. This filter works best for removing spurious peak and trough voxels and performs no averaging. A numeric value can also be supplied, which will replace all isolated or well-connected voxels.

3.95.3.7.2 Default

- `"conservative"`

3.95.3.7.3 Examples

- `"mean"`
- `"median"`
- `"conservative"`
- `"0.0"`
- `"-1.23"`
- `"1E6"`
- `"nan"`

3.95.3.8 Replace

3.95.3.8.1 Description Controls whether isolated or well-connected voxels are retained.

3.95.3.8.2 Default

- `"isolated"`

3.95.3.8.3 Supported Options

- `"isolated"`
- `"well-connected"`

3.95.3.9 NeighbourCount

3.95.3.9.1 Description Controls the number of neighbours being considered. For purposes of speed, this option is limited to specific levels of neighbour adjacency.

3.95.3.9.2 Default

- `"isolated"`

3.95.3.9.3 Examples

- `"1"`
- `"2"`
- `"3"`

3.95.3.10 AgreementCount

3.95.3.10.1 Description Controls the number of neighbours that must be in agreement for a voxel to be considered 'well-connected.'

3.95.3.10.2 Default

- `"6"`

3.95.3.10.3 Examples

- `"1"`
- `"2"`
- `"25"`

3.95.3.11 MaxDistance

3.95.3.11.1 Description The maximum distance (inclusive, in DICOM units: mm) within which neighbouring voxels will be evaluated. For spherical neighbourhoods, this distance refers to the radius. For cubic neighbourhoods, this distance refers to 'box radius' or the distance from the cube centre to the nearest point on each bounding face. Voxels separated by more than this distance will not be evaluated together.

3.95.3.11.2 Default

- `"2.0"`

3.95.3.11.3 Examples

- `"0.5"`
- `"2.0"`
- `"15.0"`

3.95.3.12 ImageSelection

3.95.3.12.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.95.3.12.2 Default

- `"last"`

3.95.3.12.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.95.3.13 DICOMMargin

3.95.3.13.1 Description The amount of margin (in the DICOM coordinate system) to spare from cropping.

3.95.3.13.2 Default

- `"0.0"`

3.95.3.13.3 Examples

- `"0.1"`
- `"2.0"`
- `"-0.5"`
- `"20.0"`

3.95.3.14 RTIMAGE

3.95.3.14.1 Description If true, attempt to crop the image using information embedded in an RTIMAGE. This option cannot be used with the other options.

3.95.3.14.2 Default

- `"true"`

3.95.3.14.3 Examples

- `"true"`
- `"false"`

3.95.3.15 ImageSelection

3.95.3.15.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.95.3.15.2 Default

- `"last"`

3.95.3.15.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.95.3.16 RowsL

3.95.3.16.1 Description The number of rows to remove, starting with the first row. Can be absolute (px), percentage (%), or distance in terms of the DICOM coordinate system. Note the DICOM coordinate system can be flipped, so the first row can be either on the top or bottom of the image.

3.95.3.16.2 Default

- `"5px"`

3.95.3.16.3 Examples

- `"0px"`
- `"10px"`
- `"100px"`
- `"15%"`
- `"15.75%"`
- `"123.45"`

3.95.3.17 RowsH

3.95.3.17.1 Description The number of rows to remove, starting with the last row. Can be absolute (px), percentage (%), or distance in terms of the DICOM coordinate system. Note the DICOM coordinate system can be flipped, so the first row can be either on the top or bottom of the image.

3.95.3.17.2 Default

- `"5px"`

3.95.3.17.3 Examples

- "0px"
- "10px"
- "100px"
- "15%"
- "15.75%"
- "123.45"

3.95.3.18 ColumnsL

3.95.3.18.1 Description The number of columns to remove, starting with the first column. Can be absolute (px), percentage (%), or distance in terms of the DICOM coordinate system. Note the DICOM coordinate system can be flipped, so the first column can be either on the top or bottom of the image.

3.95.3.18.2 Default

- "5px"

3.95.3.18.3 Examples

- "0px"
- "10px"
- "100px"
- "15%"
- "15.75%"
- "123.45"

3.95.3.19 ColumnsH

3.95.3.19.1 Description The number of columns to remove, starting with the last column. Can be absolute (px), percentage (%), or distance in terms of the DICOM coordinate system. Note the DICOM coordinate system can be flipped, so the first column can be either on the top or bottom of the image.

3.95.3.19.2 Default

- "5px"

3.95.3.19.3 Examples

- "0px"
- "10px"
- "100px"
- "15%"
- "15.75%"

- `"123.45"`

3.95.3.20 DICOMMargin

3.95.3.20.1 Description The amount of margin (in the DICOM coordinate system) to spare from cropping.

3.95.3.20.2 Default

- `"0.0"`

3.95.3.20.3 Examples

- `"0.1"`
- `"2.0"`
- `"-0.5"`
- `"20.0"`

3.95.3.21 ImageSelection

3.95.3.21.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.95.3.21.2 Default

- `"last"`

3.95.3.21.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.95.3.22 MLCModel

3.95.3.22.1 Description The MLC design geometry to use. 'VarianMillenniumMLC80' has 40 leafs in each bank; leaves are 10mm wide at isocentre; and the maximum static field size is 40cm x 40cm. 'VarianMillenniumMLC120' has 60 leafs in each bank; the 40 central leaves are 5mm wide at isocentre; the 20 peripheral leaves are 10mm wide; and the maximum static field size is 40cm x 40cm. 'VarianHD120' has 60 leafs in each bank; the 32 central leaves are 2.5mm wide at isocentre; the 28 peripheral leaves are 5mm wide; and the maximum static field size is 40cm x 22cm.

3.95.3.22.2 Default

- `"VarianMillenniumMLC120"`

3.95.3.22.3 Supported Options

- `"VarianMillenniumMLC80"`
- `"VarianMillenniumMLC120"`
- `"VarianHD120"`

3.95.3.23 MLCROILabel

3.95.3.23.1 Description An ROI imitating the MLC axes of leaf pairs is created. This is the label to apply to it. Note that the leaves are modeled with thin contour rectangles of virtually zero area. Also note that the outline colour is significant and denotes leaf pair pass/fail.

3.95.3.23.2 Default

- `"Leaves"`

3.95.3.23.3 Examples

- `"MLC_leaves"`
- `"MLC"`
- `"approx_leaf_axes"`

3.95.3.24 JunctionROILabel

3.95.3.24.1 Description An ROI imitating the junction is created. This is the label to apply to it. Note that the junctions are modeled with thin contour rectangles of virtually zero area.

3.95.3.24.2 Default

- `"Junction"`

3.95.3.24.3 Examples

- `"Junction"`
- `"Picket_Fence_Junction"`

3.95.3.25 PeakROILabel

3.95.3.25.1 Description ROIs encircling the leaf profile peaks are created. This is the label to apply to it. Note that the peaks are modeled with small squares.

3.95.3.25.2 Default

- `"Peak"`

3.95.3.25.3 Examples

- `"Peak"`
- `"Picket_Fence_Peak"`

3.95.3.26 MinimumJunctionSeparation

3.95.3.26.1 Description The minimum distance between junctions on the SAD isoplane in DICOM units (mm). This number is used to de-duplicate automatically detected junctions. Analysis results should not be sensitive to the specific value.

3.95.3.26.2 Default

- "10.0"

3.95.3.26.3 Examples

- "5.0"
- "10.0"
- "15.0"
- "25.0"

3.95.3.27 ThresholdDistance

3.95.3.27.1 Description The threshold distance in DICOM units (mm) above which MLC separations are considered to 'fail'. Each leaf pair is evaluated separately. Pass/fail status is also indicated by setting the leaf axis contour colour (blue for pass, red for fail).

3.95.3.27.2 Default

- "0.5"

3.95.3.27.3 Examples

- "0.5"
- "1.0"
- "2.0"

3.95.3.28 LeafGapsFileName

3.95.3.28.1 Description This file will contain gap and nominal-vs-actual offset distances for each leaf pair. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.95.3.28.2 Default

- ""

3.95.3.28.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.95.3.29 ResultsSummaryFileName

3.95.3.29.1 Description This file will contain a brief summary of the results. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.95.3.29.2 Default

- `""`

3.95.3.29.3 Examples

- `""`
- `"/tmp/somefile"`
- `"localfile.csv"`
- `"derivative_data.csv"`

3.95.3.30 UserComment

3.95.3.30.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data.

3.95.3.30.2 Default

- `""`

3.95.3.30.3 Examples

- `""`
- `"Using XYZ"`
- `"Patient treatment plan C"`

3.95.3.31 InteractivePlots

3.95.3.31.1 Description Whether to interactively show plots showing detected edges.

3.95.3.31.2 Default

- `"false"`

3.95.3.31.3 Examples

- `"true"`
- `"false"`

3.95.3.32 ScaleFactor

3.95.3.32.1 Description This factor is applied to the image width and height to magnify (larger than 1) or shrink (less than 1) the image. This factor only affects the output image size. Note that aspect ratio is retained, but rounding for non-integer factors may lead to small (1-2 pixel) discrepancies.

3.95.3.32.2 Default

- `"1.5"`

3.95.3.32.3 Examples

- `"0.5"`
- `"1.0"`
- `"2.0"`
- `"5.23"`

3.95.3.33 ImageFileName

3.95.3.33.1 Description The file name to use for the image. If blank, a filename will be generated sequentially.

3.95.3.33.2 Default

- `""`

3.95.3.33.3 Examples

- `""`
- `"/tmp/an_image.png"`
- `"afile.png"`

3.95.3.34 ColourMapRegex

3.95.3.34.1 Description The colour mapping to apply to the image if there is a single channel. The default will match the first available, and if there is no matching map found, the first available will be selected.

3.95.3.34.2 Default

- `".*"`

3.95.3.34.3 Supported Options

- `"Viridis"`
- `"Magma"`
- `"Plasma"`
- `"Inferno"`

- "Jet"
- "MorelandBlueRed"
- "MorelandBlackBody"
- "MorelandExtendedBlackBody"
- "KRC"
- "ExtendedKRC"
- "Kovesi_LinKRYW_5-100_c64"
- "Kovesi_LinKRYW_0-100_c71"
- "Kovesi_Cyclic_cet-c2"
- "LANLOliveGreentoBlue"
- "YgorIncandescent"
- "LinearRamp"

3.95.3.35 WindowLow

3.95.3.35.1 Description If provided, this parameter will override any existing window and level. All pixels with the intensity value or lower will be assigned the lowest possible colour according to the colour map. Not providing a valid number will disable window overrides.

3.95.3.35.2 Default

- ""

3.95.3.35.3 Examples

- ""
- "-1.23"
- "0"
- "1E4"

3.95.3.36 WindowHigh

3.95.3.36.1 Description If provided, this parameter will override any existing window and level. All pixels with the intensity value or higher will be assigned the highest possible colour according to the colour map. Not providing a valid number will disable window overrides.

3.95.3.36.2 Default

- ""

3.95.3.36.3 Examples

- ""
- "1.23"

- `"0"`
- `"10.3E4"`

3.96 ForEachDistinct

3.96.1 Description

This operation is a control flow meta-operation that partitions all available data and invokes all child operations once for each distinct partition.

3.96.2 Notes

- If this operation has no children, this operation will evaluate to a no-op.
- This operation will only partition homogeneous objects, i.e., composite objects in which all sub-objects share the same set of metadata (e.g., image arrays, since each image carries its own metadata). This guarantees there will be no side-effects due to the partitioning. For this reason, this operation is most commonly used on high-level metadata tags that are expected to be uniform across sub-objects. See the GroupImages operation to permanently partition heterogeneous image arrays.
- Each invocation is performed sequentially, and all modifications are carried forward for each grouping. However, partitions are generated before any child operations are invoked, so newly-added elements (e.g., new Image_Arrays) created by one invocation will not participate in subsequent invocations. The order of the de-partitioned data is stable, though additional elements added will follow the partition they were generated from (and will thus not necessarily be placed at the last element).
- This operation will most often be used to process data group-wise rather than as a whole.

3.96.3 Parameters

- KeysCommon

3.96.3.1 KeysCommon

3.96.3.1.1 Description Metadata keys to use for exact-match groupings on all data types. For each partition that is produced, every element will share the same key-value pair. This is generally useful for non-numeric (or integer, date, etc.) key-values. A ';'-delimited list can be specified to group on multiple criteria simultaneously. An empty string disables metadata-based grouping.

3.96.3.1.2 Default

- ""

3.96.3.1.3 Examples

- "FrameOfReferenceUID"
- "BodyPartExamined;StudyDate"
- "SeriesInstanceUID"
- "StationName"

3.97 GenerateCalibrationCurve

3.97.1 Description

This operation uses two overlapping images volumes to generate a calibration curve mapping from the first image volume to the second. Only the region within the specified ROI(s) is considered.

3.97.2 Notes

- ROI(s) are interpretted relative to the mapped-to ('reference' or 'fixed') image. The reason for this is that typically the reference images are associated with contours (e.g., planning data) and the mapped-from images do not (e.g., CBCTs that have been registered).
- This routine can handle overlapping or duplicate contours.

3.97.3 Parameters

- Channel
- ImageSelection
- RefImageSelection
- ContourOverlap
- Inclusivity
- CalibCurveFileName
- NormalizedROILabelRegex
- ROILabelRegex

3.97.3.1 Channel

3.97.3.1.1 Description The image channel to use. Zero-based. Use '-1' to operate on all available channels.

3.97.3.1.2 Default

- "-1"

3.97.3.1.3 Examples

- `"-1"`
- `"0"`
- `"1"`
- `"2"`

3.97.3.2 ImageSelection

3.97.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified. Note that these images are the 'mapped-from' or 'moving' images.

3.97.3.2.2 Default

- `"last"`

3.97.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`

- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.97.3.3 RefImageSelection

3.97.3.3.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified. Note that these images are the 'mapped-to' or 'fixed' images.

3.97.3.3.2 Default

- `"last"`

3.97.3.3.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`

- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.97.3.4 ContourOverlap

3.97.3.4.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.97.3.4.2 Default

- "ignore"

3.97.3.4.3 Supported Options

- "ignore"
- "honour_opposite_orientations"
- "overlapping_contours_cancel"
- "honour_opps"
- "overlap_cancel"

3.97.3.5 Inclusivity

3.97.3.5.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.97.3.5.2 Default

- "center"

3.97.3.5.3 Supported Options

- "center"
- "centre"
- "planar_corner_inclusive"
- "planar_inc"
- "planar_corner_exclusive"
- "planar_exc"

3.97.3.6 CalibCurveFileName

3.97.3.6.1 Description The file to which a calibration curve will be written to. The format is line-based with 4 numbers per line: (original pixel value) (uncertainty) (new pixel value) (uncertainty). Uncertainties refer to the prior number and may be uniformly zero if unknown. Lines beginning with '#' are comments. The curve is meant to be interpolated. (Later attempts to extrapolate may result in failure.)

3.97.3.6.2 Default

- `""`

3.97.3.6.3 Examples

- `"./calib.dat"`

3.97.3.7 NormalizedROILabelRegex

3.97.3.7.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.97.3.7.2 Default

- `".*"`

3.97.3.7.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.97.3.8 ROILabelRegex

3.97.3.8.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.97.3.8.2 Default

- `".*"`

3.97.3.8.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.98 GenerateSurfaceMask

3.98.1 Description

This operation generates a surface image mask, which contains information about whether each voxel is within, on, or outside the selected ROI(s).

3.98.2 Parameters

- BackgroundVal
- InteriorVal
- SurfaceVal
- NormalizedROILabelRegex
- ROILabelRegex

3.98.2.1 BackgroundVal

3.98.2.1.1 Description The value to give to voxels neither inside nor on the surface of the ROI(s).

3.98.2.1.2 Default

- `"0.0"`

3.98.2.1.3 Examples

- `"0.0"`
- `"-1.0"`
- `"1.23"`
- `"2.34E26"`

3.98.2.2 InteriorVal

3.98.2.2.1 Description The value to give to voxels within the volume of the ROI(s) but not on the surface.

3.98.2.2.2 Default

- `"1.0"`

3.98.2.2.3 Examples

- `"0.0"`
- `"-1.0"`
- `"1.23"`
- `"2.34E26"`

3.98.2.3 SurfaceVal

3.98.2.3.1 Description The value to give to voxels on the surface/boundary of ROI(s).

3.98.2.3.2 Default

- `"2.0"`

3.98.2.3.3 Examples

- `"0.0"`
- `"-1.0"`
- `"1.23"`
- `"2.34E26"`

3.98.2.4 NormalizedROILabelRegex

3.98.2.4.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.98.2.4.2 Default

- `".*"`

3.98.2.4.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.98.2.5 ROILabelRegex

3.98.2.5.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.98.2.5.2 Default

- `".*"`

3.98.2.5.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.99 GenerateSyntheticImages

3.99.1 Description

This operation generates a synthetic, regular bitmap image array. It can be used for testing how images are quantified or transformed.

3.99.2 Parameters

- NumberOfImages
- NumberOfRows
- NumberOfColumns
- NumberOfChannels
- SliceThickness
- SpacingBetweenSlices
- VoxelWidth
- VoxelHeight
- ImageAnchor
- ImagePosition
- ImageOrientationColumn
- ImageOrientationRow
- InstanceNumber
- AcquisitionNumber
- VoxelValue
- StipleValue
- Metadata

3.99.2.1 NumberOfImages

3.99.2.1.1 Description The number of images to create.

3.99.2.1.2 Default

- "100"

3.99.2.1.3 Examples

- "1"
- "100"
- "1000"

3.99.2.2 NumberOfRows

3.99.2.2.1 Description The number of rows each image should contain.

3.99.2.2.2 Default

- "256"

3.99.2.2.3 Examples

- "1"
- "100"
- "1000"

3.99.2.3 NumberOfColumns

3.99.2.3.1 Description The number of columns each image should contain.

3.99.2.3.2 Default

- "256"

3.99.2.3.3 Examples

- "1"
- "100"
- "1000"

3.99.2.4 NumberOfChannels

3.99.2.4.1 Description The number of channels each image should contain.

3.99.2.4.2 Default

- "1"

3.99.2.4.3 Examples

- "1"
- "10"
- "100"

3.99.2.5 SliceThickness

3.99.2.5.1 Description Image slices will be have this thickness (in DICOM units: mm). For most purposes, SliceThickness should be equal to SpacingBetweenSlices. If SpacingBetweenSlices is smaller than SliceThickness, images will overlap. If SpacingBetweenSlices is larger than SliceThickness, there will be a gap between images.

3.99.2.5.2 Default

- `"1.0"`

3.99.2.5.3 Examples

- `"0.1"`
- `"0.5"`
- `"1.0"`
- `"10.0"`

3.99.2.6 SpacingBetweenSlices

3.99.2.6.1 Description Image slice centres will be separated by this distance (in DICOM units: mm). For most purposes, SpacingBetweenSlices should be equal to SliceThickness. If SpacingBetweenSlices is smaller than SliceThickness, images will overlap. If SpacingBetweenSlices is larger than SliceThickness, there will be a gap between images.

3.99.2.6.2 Default

- `"1.0"`

3.99.2.6.3 Examples

- `"0.1"`
- `"0.5"`
- `"1.0"`
- `"10.0"`

3.99.2.7 VoxelWidth

3.99.2.7.1 Description Voxels will have this (in-plane) width (in DICOM units: mm). This means that row-adjacent voxels centres will be separated by VoxelWidth). Each voxel will have dimensions: VoxelWidth x VoxelHeight x SliceThickness.

3.99.2.7.2 Default

- `"1.0"`

3.99.2.7.3 Examples

- `"0.1"`
- `"0.5"`
- `"1.0"`
- `"10.0"`

3.99.2.8 VoxelHeight

3.99.2.8.1 Description Voxels will have this (in-plane) height (in DICOM units: mm). This means that column-adjacent voxels centres will be separated by VoxelHeight). Each voxel will have dimensions: VoxelWidth x VoxelHeight x SliceThickness.

3.99.2.8.2 Default

- `"1.0"`

3.99.2.8.3 Examples

- `"0.1"`
- `"0.5"`
- `"1.0"`
- `"10.0"`

3.99.2.9 ImageAnchor

3.99.2.9.1 Description A point in 3D space which denotes the origin (in DICOM units: mm). All other vectors are taken to be relative to this point. Under most circumstance the anchor should be (0,0,0). Specify coordinates separated by commas.

3.99.2.9.2 Default

- `"0.0, 0.0, 0.0"`

3.99.2.9.3 Examples

- `"0.0, 0.0, 0.0"`
- `"0.0,0.0,0.0"`
- `"1.0, -2.3, 45.6"`

3.99.2.10 ImagePosition

3.99.2.10.1 Description The centre of the row=0, column=0 voxel in the first image (in DICOM units: mm). Specify coordinates separated by commas.

3.99.2.10.2 Default

- `"0.0, 0.0, 0.0"`

3.99.2.10.3 Examples

- `"0.0, 0.0, 0.0"`
- `"100.0,100.0,100.0"`
- `"1.0, -2.3, 45.6"`

3.99.2.11 ImageOrientationColumn

3.99.2.11.1 Description The orientation unit vector that is aligned with image columns. Care should be taken to ensure ImageOrientationRow and ImageOrientationColumn are orthogonal. (A Gram-Schmidt orthogonalization procedure ensures they are, but the image orientation may not match the expected orientation.) Note that the magnitude will also be scaled to unit length for convenience. Specify coordinates separated by commas.

3.99.2.11.2 Default

- `"1.0, 0.0, 0.0"`

3.99.2.11.3 Examples

- `"1.0, 0.0, 0.0"`
- `"1.0, 1.0, 0.0"`
- `"0.0, 0.0, -1.0"`

3.99.2.12 ImageOrientationRow

3.99.2.12.1 Description The orientation unit vector that is aligned with image rows. Care should be taken to ensure ImageOrientationRow and ImageOrientationColumn are orthogonal. (A Gram-Schmidt orthogonalization procedure ensures they are, but the image orientation may not match the expected orientation.) Note that the magnitude will also be scaled to unit length for convenience. Specify coordinates separated by commas.

3.99.2.12.2 Default

- `"0.0, 1.0, 0.0"`

3.99.2.12.3 Examples

- `"0.0, 1.0, 0.0"`
- `"0.0, 1.0, 1.0"`
- `"-1.0, 0.0, 0.0"`

3.99.2.13 InstanceNumber

3.99.2.13.1 Description A number affixed to the first image, and then incremented and affixed for each subsequent image.

3.99.2.13.2 Default

- `"1"`

3.99.2.13.3 Examples

- `"1"`
- `"100"`
- `"1234"`

3.99.2.14 AcquisitionNumber

3.99.2.14.1 Description A number affixed to all images, meant to indicate membership in a single acquisition.

3.99.2.14.2 Default

- `"1"`

3.99.2.14.3 Examples

- `"1"`
- `"100"`
- `"1234"`

3.99.2.15 VoxelValue

3.99.2.15.1 Description The value that is assigned to all voxels, or possibly every other voxel. Note that if StipleValue is given a finite value, only half the voxels will be assigned a value of VoxelValue and the other half will be assigned a value of StipleValue. This produces a checkerboard pattern.

3.99.2.15.2 Default

- `"0.0"`

3.99.2.15.3 Examples

- `"0.0"`
- `"1.0E4"`
- `"-1234"`
- `"nan"`

3.99.2.16 StipleValue

3.99.2.16.1 Description The value that is assigned to every other voxel. If StipleValue is given a finite value, half of all voxels will be assigned a value of VoxelValue and the other half will be assigned a value of StipleValue. This produces a checkerboard pattern.

3.99.2.16.2 Default

- `"nan"`

3.99.2.16.3 Examples

- `"1.0"`
- `"-1.0E4"`
- `"1234"`

3.99.2.17 Metadata

3.99.2.17.1 Description A semicolon-separated list of 'key@value' metadata to imbue into each image. This metadata will overwrite any existing keys with the provided values.

3.99.2.17.2 Default

- `""`

3.99.2.17.3 Examples

- `"keyA@valueA;keyB@valueB"`

3.100 GenerateVirtualDataContourViaThresholdTestV1

3.100.1 Description

This operation generates data suitable for testing the ContourViaThreshold operation.

3.100.2 Parameters

No registered options.

3.101 GenerateVirtualDataDoseStairsV1

3.101.1 Description

This operation generates a dosimetric stairway. It can be used for testing how dosimetric data is transformed.

3.101.2 Parameters

No registered options.

3.102 GenerateVirtualDataImageSphereV1

3.102.1 Description

This operation generates a bitmap image of a sphere. It can be used for testing how images are quantified or transformed.

3.102.2 Parameters

No registered options.

3.103 GenerateVirtualDataPerfusionV1

3.103.1 Description

This operation generates data suitable for testing perfusion modeling operations. There are no specific checks in this code. Another operation performs the actual validation. You might be able to manually verify if the perfusion model admits a simple solution.

3.103.2 Parameters

No registered options.

3.104 GenerateWarp

3.104.1 Description

This operation can be used to create a transformation object. The transformation object can later be applied to objects with spatial extent.

3.104.2 Parameters

- Transforms
- TransformLabel
- Metadata

3.104.2.1 Transforms

3.104.2.1.1 Description This parameter is used to specify one or more transformations. Current primitives include translation, scaling, mirroring, and rotation. • Translations have three configurable scalar parameters denoting the translation along x, y, and z in the DICOM coordinate system. Translating $x = 1.0$, $y = -2.0$, and $z = 0.3$ can be specified as 'translate(1.0, -2.0, 0.3)'. • The scale (actually 'homothetic') transformation has four configurable scalar parameters denoting the scale centre 3-vector and the magnification factor. Note that the magnification factor can be negative, which will cause the mesh to be inverted along x, y, and z axes and magnified. Take note that face orientations will also become inverted. Magnifying by 2.7x about $(1.23, -2.34, 3.45)$ can be specified as 'scale(1.23, -2.34, 3.45, 2.7)'. A standard scale transformation can be achieved by taking the centre to be the origin. • The mirror transformation has six configurable scalar parameters denoting an oriented plane about which a mirror is performed. Mirroring in the plane that intersects $(1, 2, 3)$ and has a normal toward $(1, 0, 0)$ can be specified as 'mirror(1,2,3, 1,0,0)'. • Rotations around an arbitrary axis line can be accomplished. The rotation transformation has seven configurable scalar parameters denoting the rotation centre 3-vector, the rotation axis 3-vector, and the rotation angle in radians. A rotation of pi radians around the axis line parallel to vector $(1.0, 0.0, 0.0)$ that intersects the point $(4.0, 5.0, 6.0)$ can be specified as 'rotate(4.0, 5.0, 6.0, 1.0, 0.0, 0.0, 3.141592653)'. • A transformation can be composed of one or more primitive transformations applied sequentially. Primitives can be separated by a ';' and are evaluated from left to right.

3.104.2.1.2 Default

- `"translate(0.0, 0.0, 0.0)"`

3.104.2.1.3 Examples

- `"translate(1.0, -2.0, 0.3)"`
- `"scale(1.23, -2.34, 3.45, 2.7)"`

- "mirror(0,0,0, 1,0,0)"
- "rotate(4.0, 5.0, 6.0, 1.0, 0.0, 0.0, 3.141592653)"
- "translate(1,0,0) ; scale(0,0,0, 5) ; translate(-1,0,0)"

3.104.2.2 TransformLabel

3.104.2.2.1 Description A label to attach to the transformation.

3.104.2.2.2 Default

- "unspecified"

3.104.2.2.3 Examples

- "unspecified"
- "offset"
- "expansion"
- "rotation_around_xyz"
- "move_to_origin"

3.104.2.3 Metadata

3.104.2.3.1 Description A semicolon-separated list of 'key@value' metadata to imbue into the transform. This metadata will overwrite any existing keys with the provided values.

3.104.2.3.2 Default

- ""

3.104.2.3.3 Examples

- "keyA@valueA;keyB@valueB"

3.105 GiveWholeImageArrayABoneWindowLevel

3.105.1 Description

This operation runs the images in an image array through a uniform window-and-leveler instead of per-slice window-and-level or no window-and-level at all. Data is modified and no copy is made!

3.105.2 Parameters

No registered options.

3.106 GiveWholeImageArrayAHeadAndNeckWindowLevel

3.106.1 Description

This operation runs the images in an image array through a uniform window-and-leveler instead of per-slice window-and-level or no window-and-level at all. Data is modified and no copy is made!

3.106.2 Parameters

No registered options.

3.107 GiveWholeImageArrayAThoraxWindowLevel

3.107.1 Description

This operation runs the images in an image array through a uniform window-and-leveler instead of per-slice window-and-level or no window-and-level at all. Data is modified and no copy is made!

3.107.2 Parameters

No registered options.

3.108 GiveWholeImageArrayAnAbdominalWindowLevel

3.108.1 Description

This operation runs the images in an image array through a uniform window-and-leveler instead of per-slice window-and-level or no window-and-level at all. Data is modified and no copy is made!

3.108.2 Parameters

No registered options.

3.109 GiveWholeImageArrayAnAlphaBetaWindowLevel

3.109.1 Description

This operation runs the images in an image array through a uniform window-and-leveler instead of per-slice window-and-level or no window-and-level at all. Data is modified and no copy is made!

3.109.2 Parameters

No registered options.

3.110 GridBasedRayCastDoseAccumulate

3.110.1 Description

This operation performs a ray casting to estimate the surface dose of an ROI.

3.110.2 Parameters

- DoseMapFileName
- DoseLengthMapFileName
- LengthMapFileName
- NormalizedReferenceROILabelRegex
- NormalizedROILabelRegex
- ReferenceROILabelRegex
- ROILabelRegex
- SmallestFeature
- RaydL
- GridRows
- GridColumns
- SourceDetectorRows
- SourceDetectorColumns
- NumberOfImages

3.110.2.1 DoseMapFileName

3.110.2.1.1 Description A filename (or full path) for the dose image map. Note that this file is approximate, and may not be accurate. There is more information available when you use the length and dose*length maps instead. However, this file is useful for viewing and eyeballing tuning settings. The format is FITS. Leave empty to dump to generate a unique temporary file.

3.110.2.1.2 Default

- `""`

3.110.2.1.3 Examples

- `""`
- `"/tmp/dose.fits"`
- `"localfile.fits"`
- `"derivative_data.fits"`

3.110.2.2 DoseLengthMapFileName

3.110.2.2.1 Description A filename (or full path) for the (dose)*(length traveled through the ROI peel) image map. The format is FITS. Leave empty to dump to generate a unique temporary file.

3.110.2.2.2 Default

- ""

3.110.2.2.3 Examples

- ""
- "/tmp/doselength.fits"
- "localfile.fits"
- "derivative_data.fits"

3.110.2.3 LengthMapFileName

3.110.2.3.1 Description A filename (or full path) for the (length traveled through the ROI peel) image map. The format is FITS. Leave empty to dump to generate a unique temporary file.

3.110.2.3.2 Default

- ""

3.110.2.3.3 Examples

- ""
- "/tmp/surfacelength.fits"
- "localfile.fits"
- "derivative_data.fits"

3.110.2.4 NormalizedReferenceROILabelRegex

3.110.2.4.1 Description A regex matching reference ROI labels/names to consider. The default will match all available ROIs, which is non-sensical. The reference ROI is used to orient the cleaving plane to trim the grid surface mask.

3.110.2.4.2 Default

- ".*"

3.110.2.4.3 Examples

- ".*"
- ".*Prostate.*"
- "Left Kidney"
- "Gross Liver"

3.110.2.5 NormalizedROILabelRegex

3.110.2.5.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.110.2.5.2 Default

- ".*"

3.110.2.5.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.110.2.6 ReferenceROILabelRegex

3.110.2.6.1 Description A regex matching reference ROI labels/names to consider. The default will match all available ROIs, which is non-sensical. The reference ROI is used to orient the cleaving plane to trim the grid surface mask.

3.110.2.6.2 Default

- ".*"

3.110.2.6.3 Examples

- ".*"
- ".*[pP]rostate.*"
- "body"
- "Gross_Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.110.2.7 ROILabelRegex

3.110.2.7.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.110.2.7.2 Default

- ".*"

3.110.2.7.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.110.2.8 SmallestFeature

3.110.2.8.1 Description A length giving an estimate of the smallest feature you want to resolve. Quantity is in the DICOM coordinate system.

3.110.2.8.2 Default

- "0.5"

3.110.2.8.3 Examples

- "1.0"
- "2.0"
- "0.5"
- "5.0"

3.110.2.9 RaydL

3.110.2.9.1 Description The distance to move a ray each iteration. Should be « img_thickness and « cylinder_radius. Making too large will invalidate results, causing rays to pass through the surface without registering any dose accumulation. Making too small will cause the run-time to grow and may eventually lead to truncation or round-off errors. Quantity is in the DICOM coordinate system.

3.110.2.9.2 Default

- "0.1"

3.110.2.9.3 Examples

- "0.1"
- "0.05"
- "0.01"
- "0.005"

3.110.2.10 GridRows

3.110.2.10.1 Description The number of rows in the surface mask grid images.

3.110.2.10.2 Default

- "512"

3.110.2.10.3 Examples

- "10"
- "50"
- "128"
- "1024"

3.110.2.11 GridColumns

3.110.2.11.1 Description The number of columns in the surface mask grid images.

3.110.2.11.2 Default

- "512"

3.110.2.11.3 Examples

- "10"
- "50"
- "128"
- "1024"

3.110.2.12 SourceDetectorRows

3.110.2.12.1 Description The number of rows in the resulting images. Setting too fine relative to the surface mask grid or dose grid is futile.

3.110.2.12.2 Default

- "1024"

3.110.2.12.3 Examples

- "10"
- "50"
- "128"
- "1024"

3.110.2.13 SourceDetectorColumns

3.110.2.13.1 Description The number of columns in the resulting images. Setting too fine relative to the surface mask grid or dose grid is futile.

3.110.2.13.2 Default

- "1024"

3.110.2.13.3 Examples

- "10"
- "50"
- "128"
- "1024"

3.110.2.14 NumberOfImages

3.110.2.14.1 Description The number of images used for grid-based surface detection. Leave negative for computation of a reasonable value; set to something specific to force an override.

3.110.2.14.2 Default

- `"-1"`

3.110.2.14.3 Examples

- `"-1"`
- `"10"`
- `"50"`
- `"100"`

3.111 GroupImages

3.111.1 Description

This operation will group individual image slices into partitions (Image_Arrays) based on the values of the specified metadata tags. DICOMautomaton operations are usually performed on containers rather than individual images, and grouping can express connections between images. For example a group could contain the scans belonging to a whole study, one of the series in a study, individual image volumes within a single series (e.g., a 3D volume in a temporal perfusion scan), or individual slices. A group could also contain all the slices that intersect a given plane, or were taken on a specified StationName.

3.111.2 Notes

- Images are moved, not copied.
- Image order within a group is retained (i.e., stable grouping), but groups are appended to the back of the Image_Array list according to the default sort for the group's key-value value.
- Images that do not contain the specified metadata will be grouped into a special N/A group at the end.

3.111.3 Parameters

- ImageSelection
- KeysCommon
- AutoSelectKeysCommon
- Enforce

3.111.3.1 ImageSelection

3.111.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.111.3.1.2 Default

- `"all"`

3.111.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.111.3.2 KeysCommon

3.111.3.2.1 Description Image metadata keys to use for exact-match groupings. For each group that is produced, every image will share the same key-value

pair. This is generally useful for non-numeric (or integer, date, etc.) key-values. A ';'-delimited list can be specified to group on multiple criteria simultaneously. An empty string disables metadata-based grouping.

3.111.3.2.2 Default

- `""`

3.111.3.2.3 Examples

- `"SeriesNumber"`
- `"BodyPartExamined;StudyDate"`
- `"SeriesInstanceUID"`
- `"StationName"`

3.111.3.3 AutoSelectKeysCommon

3.111.3.3.1 Description Attempt to automatically select the single image metadata key that partitions images into approximately evenly-sized partitions. Currently, some basic and broad assumptions are used to filter candidate keys. The criteria will not work in all cases, but might help identify candidates. This option cannot be enabled when providing the KeysCommon parameter.

3.111.3.3.2 Default

- `"false"`

3.111.3.3.3 Supported Options

- `"true"`
- `"false"`

3.111.3.4 Enforce

3.111.3.4.1 Description Other specialized grouping operations that involve custom logic. Currently, only 'no-overlap' is available, but it has two variants. Both partition based on the spatial extent of images; in each non-overlapping partition, no two images will spatially overlap. 'No-overlap-as-is' will effectively insert partitions without altering the order. A partition is inserted when an image is found to overlap with an image already within the partition. For this grouping to be useful, images must be sorted so that partitions can be inserted without any necessary reordering. An example of when this grouping is useful is CT shuttling in which the ordering of images alternate between increasing and decreasing SliceNumber. Note that, depending on the ordering, some partitions may therefore be incomplete. 'No-overlap-adjust' will rearrange images so that the first partition is always complete. This is achieved by building a queue

of spatially-overlapping images and greedily stealing one of each kind when constructing partitions. An example of when this grouping is useful are 4DCTs which have been acquired for all phases while the couch remains at a single SliceNumber; the images are ordered on disk in the acquisition order (i.e., alike SliceNumbers are bunched together) but the logical analysis order is that each contiguous volume should represent a single phase. An empty string disables logic-based grouping.

3.111.3.4.2 Default

- `""`

3.111.3.4.3 Supported Options

- `"no-overlap-as-is"`
- `"no-overlap-adjust"`

3.112 GrowContours

3.112.1 Description

This routine will grow (or shrink) 2D contours in their plane by the specified amount. Growth is accomplish by translating vertices away from the interior by the specified amount. The direction is chosen to be the direction opposite of the in-plane normal produced by averaging the line segments connecting the contours.

3.112.2 Parameters

- NormalizedROILabelRegex
- ROILabelRegex
- Distance

3.112.2.1 NormalizedROILabelRegex

3.112.2.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful

for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.112.2.1.2 Default

- `".*"`

3.112.2.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.112.2.2 ROILabelRegex

3.112.2.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.112.2.2.2 Default

- `".*"`

3.112.2.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.112.2.3 Distance

3.112.2.3.1 Description The distance to translate contour vertices. (The direction is outward.)

3.112.2.3.2 Default

- `"0.00354165798657632"`

3.112.2.3.3 Examples

- `"1E-5"`
- `"0.321"`
- `"1.1"`
- `"15.3"`

3.113 HighlightROIs

3.113.1 Description

This operation overwrites voxel data inside and/or outside of ROI(s) to create an image representation of a set of contours. It can handle overlapping or duplicate contours.

3.113.2 Notes

- The 'receding_squares' implementation uses a simplistic one-pass approach that only considers only the immediate left and immediate up neighbours to determine the necessary intensity of the (*this) voxel. The intensity is over-specified, so in general will result in the exact intensity needed to exactly reproduce the original contours. Slight differences can arise do to averaging and numerical imprecision, especially if the input comes from a marching algorithm (common!) which can result in geometrical alignment and degenerate voxel inclusions. The 'receding_squares' implementation was developed with the expectations that: (1) the entire image will be overwritten, (2) contours are accurate and selective, so that ContourOverlap should be either 'honour_opposite_orientations' or 'overlapping_contours_cancel', and that (3) the contour detail and image grid resolution are sufficiently matched that it is uncommon for multiple contours to pass between adjacent voxels. For expectation (2), using 'overlapping_contours_cancel' produces the best results, since all contours will be recreated as much as possible. Expectation (3) could significantly impact round-trip contour accuracy, so consider using high-resolution images and, if possible, avoid pathological contours (e.g., multiple colinear contours separated by small distances).
- The 'receding_squares' method works best when all values (interior and exterior) can be overwritten. This affords the most control and gives the most accurate results. If some values cannot be overwritten, the algorithm will try to account for the loss of freedom, but may be too constrained. If this is necessary, consider providing a large voxel value range.

- Inclusivity option does not apply to the 'receding_squares' method.
- Neither 'receding_squares' nor 'binary' methods require InteriorVal and ExteriorVal to be ordered.

3.113.3 Parameters

- Channel
- ImageSelection
- ContourOverlap
- Inclusivity
- Method
- ExteriorVal
- InteriorVal
- ExteriorOverwrite
- InteriorOverwrite
- NormalizedROILabelRegex
- ROILabelRegex

3.113.3.1 Channel

3.113.3.1.1 Description The image channel to use. Zero-based. Use '-1' to operate on all available channels.

3.113.3.1.2 Default

- `"-1"`

3.113.3.1.3 Examples

- `"-1"`
- `"0"`
- `"1"`
- `"2"`

3.113.3.2 ImageSelection

3.113.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.113.3.2.2 Default

- `"last"`

3.113.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.113.3.3 ContourOverlap

3.113.3.3.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. This will effectively honour only the outermost contour regardless of orientation, but provides the most predictable and consistent results. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. This is useful for Boolean structures where contour orientation is significant for interior contours (holes). If contours do not have consistent overlap (e.g., if contours intersect) the results can be unpredictable and hard to interpret. The option 'overlapping_contours_cancel' ignores orientation and alternately cancerls all overlapping contours. Again, if the contours do not have consistent overlap (e.g., if contours intersect) the results can be unpredictable and hard to interpret.

3.113.3.3.2 Default

- `"ignore"`

3.113.3.3.3 Supported Options

- `"ignore"`
- `"honour_opposite_orientations"`
- `"overlapping_contours_cancel"`
- `"honour_opps"`
- `"overlap_cancel"`

3.113.3.4 Inclusivity

3.113.3.4.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.113.3.4.2 Default

- `"center"`

3.113.3.4.3 Supported Options

- `"center"`
- `"centre"`
- `"planar_corner_inclusive"`
- `"planar_inc"`
- `"planar_corner_exclusive"`
- `"planar_exc"`

3.113.3.5 Method

3.113.3.5.1 Description Controls the type of image mask that is generated. The default, 'binary', exclusively overwrites voxels with the InteriorValue or ExteriorValue. Another method is 'receding_squares' which creates a mask which, if processed with the marching-squares algorithm, will (mostly) recreate the original contours. The 'receding_squares' can be considered the inverse of the marching-squares algorithm. Note that the 'receding_squares' implementation is not optimized for speed.

3.113.3.5.2 Default

- `"binary"`

3.113.3.5.3 Supported Options

- `"binary"`
- `"receding_squares"`

3.113.3.6 ExteriorVal

3.113.3.6.1 Description The value to give to voxels outside the specified ROI(s). For the 'binary' method, note that this value will be ignored if exterior overwrites are disabled. For the 'receding_squares' method this value is used to define the threshold needed to recover the original contours (mean of InteriorVal and ExteriorVal).

3.113.3.6.2 Default

- `"0.0"`

3.113.3.6.3 Examples

- `"0.0"`
- `"-1.0"`
- `"1.23"`
- `"2.34E26"`

3.113.3.7 InteriorVal

3.113.3.7.1 Description The value to give to voxels within the specified ROI(s). For the 'binary' method, note that this value will be ignored if interior overwrites are disabled. For the 'receding_squares' method this value is used to define the threshold needed to recover the original contours (mean of InteriorVal and ExteriorVal).

3.113.3.7.2 Default

- `"1.0"`

3.113.3.7.3 Examples

- `"0.0"`
- `"-1.0"`
- `"1.23"`
- `"2.34E26"`

3.113.3.8 ExteriorOverwrite

3.113.3.8.1 Description Whether to overwrite voxels exterior to the specified ROI(s).

3.113.3.8.2 Default

- `"true"`

3.113.3.8.3 Examples

- `"true"`
- `"false"`

3.113.3.9 InteriorOverwrite

3.113.3.9.1 Description Whether to overwrite voxels interior to the specified ROI(s).

3.113.3.9.2 Default

- `"true"`

3.113.3.9.3 Examples

- `"true"`
- `"false"`

3.113.3.10 NormalizedROILabelRegex

3.113.3.10.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.113.3.10.2 Default

- `".*"`

3.113.3.10.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.113.3.11 ROILabelRegex

3.113.3.11.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.113.3.11.2 Default

- ".*"

3.113.3.11.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.114 ImageRoutineTests

3.114.1 Description

This operation performs a series of sub-operations that are generally useful when inspecting an image.

3.114.2 Parameters

No registered options.

3.115 ImprintImages

3.115.1 Description

This operation creates imprints of point clouds on the selected images. Images are modified where the points are coindicident.

3.115.2 Parameters

- ImageSelection
- PointSelection
- VoxelValue
- Channel

3.115.2.1 ImageSelection

3.115.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.115.2.1.2 Default

- `"last"`

3.115.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.115.2.2 PointSelection

3.115.2.2.1 Description Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.115.2.2.2 Default

- `"last"`

3.115.2.2.3 Examples

- "last"
- "first"
- "all"
- "none"
- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.115.2.3 VoxelValue

3.115.2.3.1 Description The value to give voxels which are coincident with a point from the point cloud. Note that point cloud attributes, if present, may override this value.

3.115.2.3.2 Default

- "1.0"

3.115.2.3.3 Examples

- "-1.0"
- "0.0"
- "1.23"
- "nan"
- "inf"

3.115.2.4 Channel

3.115.2.4.1 Description The image channel to use. Zero-based.

3.115.2.4.2 Default

- "0"

3.115.2.4.3 Examples

- "0"
- "1"
- "2"

3.116 InterpolateSlices

3.116.1 Description

This operation interpolates the slices of an image array using a reference image array, effectively performing trilinear interpolation. This operation is meant to prepare image arrays to be compared or operated on in a per-voxel manner.

3.116.2 Notes

- No images are overwritten by this operation. The outgoing images will inherit (interpolated) voxel values from the selected images and image geometry from the reference images.
- If all images (selected and reference, altogether) are detected to be rectilinear, this operation will avoid in-plane interpolation and will thus be much faster. There is no **need** for rectilinearity, however without it sections of the image that cannot reasonably be interpolated (via plane-orthogonal projection onto the reference images) will be invalid and marked with NaNs. Non-rectilearity which amounts to a differing number of rows or columns will merely be slower to interpolate.

3.116.3 Parameters

- ImageSelection
- ReferenceImageSelection
- Channel

3.116.3.1 ImageSelection

3.116.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note

that ‘!numerous’ means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., ‘fewest’).

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ‘;’ and are applied in the order specified.

3.116.3.1.2 Default

- `"all"`

3.116.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.116.3.2 ReferenceImageSelection

3.116.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or ‘4D’ time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be ‘first’, ‘last’, ‘none’, or ‘all’ literals. Additionally ‘#N’ for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, ‘#-N’ selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a ‘!’.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a ‘!’). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the ‘numerous’ and ‘fewest’ literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a ‘!’. Note that ‘!numerous’ means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., ‘fewest’).

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.116.3.2.2 Default

- `"all"`

3.116.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.116.3.3 Channel

3.116.3.3.1 Description The channel to compare (zero-based). A negative value will result in all channels being interpolated, otherwise unspecified channels are merely default initialized. Note that both test images and reference images will share this specifier.

3.116.3.3.2 Default

- `"-1"`

3.116.3.3.3 Examples

- `"-1"`
- `"0"`
- `"1"`
- `"2"`

3.117 IsolatedVoxelFilter

3.117.1 Description

This routine applies a filter that discriminates between well-connected and isolated voxels. Isolated voxels can either be filtered out or retained. This

operation considers the 3D neighbourhood surrounding a voxel.

3.117.2 Notes

- The provided image collection must be rectilinear.
- If the neighbourhood involves voxels that do not exist, they are treated as NaNs in the same way that voxels with the NaN value are treated.

3.117.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Channel
- Replacement
- Replace
- NeighbourCount
- AgreementCount
- MaxDistance

3.117.3.1 ImageSelection

3.117.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.117.3.1.2 Default

- `"last"`

3.117.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.117.3.2 NormalizedROILabelRegex

3.117.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.117.3.2.2 Default

- `".*"`

3.117.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.117.3.3 ROILabelRegex

3.117.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.117.3.3.2 Default

- `".*"`

3.117.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.117.3.4 Channel

3.117.3.4.1 Description The channel to operated on (zero-based). Negative values will cause all channels to be operated on.

3.117.3.4.2 Default

- `"0"`

3.117.3.4.3 Examples

- `"-1"`
- `"0"`
- `"1"`

3.117.3.5 Replacement

3.117.3.5.1 Description Controls how replacements are generated. 'Mean' and 'median' replacement strategies replace the voxel value with the mean and median, respectively, from the surrounding neighbourhood. 'Conservative' refers to the so-called conservative filter that suppresses isolated peaks; for every voxel considered, the voxel intensity is clamped to the local neighbourhood's extrema. This filter works best for removing spurious peak and trough voxels and performs no averaging. A numeric value can also be supplied, which will replace all isolated or well-connected voxels.

3.117.3.5.2 Default

- `"mean"`

3.117.3.5.3 Examples

- `"mean"`
- `"median"`
- `"conservative"`
- `"0.0"`
- `"-1.23"`
- `"1E6"`
- `"nan"`

3.117.3.6 Replace

3.117.3.6.1 Description Controls whether isolated or well-connected voxels are retained.

3.117.3.6.2 Default

- `"isolated"`

3.117.3.6.3 Supported Options

- `"isolated"`
- `"well-connected"`

3.117.3.7 NeighbourCount

3.117.3.7.1 Description Controls the number of neighbours being considered. For purposes of speed, this option is limited to specific levels of neighbour adjacency.

3.117.3.7.2 Default

- `"isolated"`

3.117.3.7.3 Examples

- `"1"`
- `"2"`
- `"3"`

3.117.3.8 AgreementCount

3.117.3.8.1 Description Controls the number of neighbours that must be in agreement for a voxel to be considered 'well-connected.'

3.117.3.8.2 Default

- `"6"`

3.117.3.8.3 Examples

- `"1"`
- `"2"`
- `"25"`

3.117.3.9 MaxDistance

3.117.3.9.1 Description The maximum distance (inclusive, in DICOM units: mm) within which neighbouring voxels will be evaluated. For spherical neighbourhoods, this distance refers to the radius. For cubic neighbourhoods, this distance refers to 'box radius' or the distance from the cube centre to the nearest point on each bounding face. Voxels separated by more than this distance will not be evaluated together.

3.117.3.9.2 Default

- `"2.0"`

3.117.3.9.3 Examples

- `"0.5"`
- `"2.0"`
- `"15.0"`

3.118 LoadFiles

3.118.1 Description

This operation loads files on-the-fly.

3.118.2 Notes

- This operation requires all files provided to it to exist and be accessible. Inaccessible files are not silently ignored and will cause this operation to fail.

3.118.3 Parameters

- FileName

3.118.3.1 FileName

3.118.3.1.1 Description This file will be parsed and loaded. All file types supported by the DICOMautomaton system can be loaded in this way. Currently this includes serialized Drover class files, DICOM files, FITS image files, and XYZ point cloud files.

3.118.3.1.2 Default

- `""`

3.118.3.1.3 Examples

- `"/tmp/image.dcm"`
- `"rois.dcm"`
- `"dose.dcm"`
- `"image.fits"`
- `"point_cloud.xyz"`

3.119 LogScale

3.119.1 Description

This operation log-scales pixels for all available image arrays. This functionality is often desired for viewing purposes, to make the pixel level changes appear more linear. Be weary of using for anything quantitative!

3.119.2 Parameters

- ImageSelection

3.119.2.1 ImageSelection

3.119.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.119.2.1.2 Default

- `"last"`

3.119.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.120 MakeMeshesManifold

3.120.1 Description

This operation attempts to make non-manifold surface meshes into manifold meshes. This operation is needed for operations requiring meshes to represent polyhedra.

3.120.2 Notes

- This routine will invalidate any imbued special attributes from the original mesh.
- It may not be possible to accomplish manifold-ness.
- Mesh features (vertices, faces, edges) may disappear in this routine.

3.120.3 Parameters

- MeshLabel
- MeshSelection

3.120.3.1 MeshLabel

3.120.3.1.1 Description A label to attach to the new manifold mesh.

3.120.3.1.2 Default

- `"unspecified"`

3.120.3.1.3 Examples

- `"unspecified"`
- `"body"`
- `"air"`
- `"bone"`
- `"invalid"`
- `"above_zero"`
- `"below_5.3"`

3.120.3.2 MeshSelection

3.120.3.2.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be ‘first’, ‘last’, ‘none’, or ‘all’ literals. Additionally ‘#N’ for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, ‘#-N’ selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a ‘!’.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a ‘!’). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the ‘numerous’ and ‘fewest’ literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a ‘!’. Note that ‘!numerous’ means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., ‘fewest’).

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ‘;’ and are applied in the order specified.

3.120.3.2.2 Default

- `"last"`

3.120.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.121 MaxMinPixels

3.121.1 Description

This operation replaces pixels with the pixel-wise difference (max)-(min).

3.121.2 Parameters

No registered options.

3.122 MeldDose

3.122.1 Description

This operation melds all available dose image data. At a high level, dose melding sums overlapping pixel values for multi-part dose arrays. For more information about what this specifically entails, refer to the appropriate subroutine.

3.122.2 Parameters

No registered options.

3.123 MinkowskiSum3D

3.123.1 Description

This operation computes a Minkowski sum or symmetric difference of a 3D surface mesh generated from the selected ROIs with a sphere. The effect is that a margin is added or subtracted to the ROIs, causing them to 'grow' outward or 'shrink' inward. Exact and inexact routines can be used.

3.123.2 Parameters

- NormalizedROILabelRegex
- ROILabelRegex
- ImageSelection
- Operation
- Distance

3.123.2.1 NormalizedROILabelRegex

3.123.2.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.123.2.1.2 Default

- ".*"

3.123.2.1.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.123.2.2 ROILabelRegex

3.123.2.2.1 Description A regex matching ROI labels/names to consider. The default will match all available ROIs. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regex to avoid them. All ROIs have to match the single regex, so use the 'or' token if needed. Regex is case insensitive and uses grep syntax.

3.123.2.2.2 Default

- ".*"

3.123.2.2.3 Examples

- ".*"
- ".*body.*"
- "body"
- "Gross_Liver"
- ".*parotid.*|.*sub.*mand.*"
- "left_parotid|right_parotid|eyes"

3.123.2.3 ImageSelection

3.123.2.3.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified. Note that the selected images are used to sample the new contours on. Image planes need not match the original since a full 3D mesh surface is generated.

3.123.2.3.2 Default

- `"last"`

3.123.2.3.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.123.2.4 Operation

3.123.2.4.1 Description The specific operation to perform. Available options are: 'dilate_exact_surface', 'dilate_exact_vertex', 'dilate_inexact_isotropic', 'erode_inexact_isotropic', and 'shell_inexact_isotropic'.

3.123.2.4.2 Default

- `"dilate_inexact_isotropic"`

3.123.2.4.3 Supported Options

- `"dilate_exact_surface"`
- `"dilate_exact_vertex"`
- `"dilate_inexact_isotropic"`
- `"erode_inexact_isotropic"`
- `"shell_inexact_isotropic"`

3.123.2.5 Distance

3.123.2.5.1 Description For dilation and erosion operations, this parameter controls the distance the surface should travel. For shell operations, this parameter controls the resultant thickness of the shell. In all cases DICOM units are assumed.

3.123.2.5.2 Default

- `"1.0"`

3.123.2.5.3 Examples

- `"0.5"`
- `"1.0"`
- `"2.0"`
- `"3.0"`
- `"5.0"`

3.124 ModelIVIM

3.124.1 Description

This operation fits an Intra-voxel Incoherent Motion (IVIM) model to a series of diffusion-weighted MR images.

3.124.2 Notes

- Images are overwritten, but their geometry is used to define the final map. ReferenceImages are used for modeling, but are treated as read-only. ReferenceImages should correspond to unique b-values, one b-value per ReferenceImages array.
- The reference image array must be rectilinear. (This is a requirement specific to this implementation, a less restrictive implementation could overcome the issue.)
- For the fastest and most accurate results, test and reference image arrays should spatially align. However, alignment is **not** necessary. If test and

reference image arrays are aligned, image adjacency can be precomputed and the analysis will be faster. If not, image adjacency must be evaluated for each image slice. If this also fails, it will be evaluated for every voxel.

- This operation will make use of interpolation if corresponding voxels do not exactly overlap.

3.124.3 Parameters

- ImageSelection
- ReferenceImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Model
- Channel
- TestImgLowerThreshold
- TestImgUpperThreshold

3.124.3.1 ImageSelection

3.124.3.1.1 Description The transformed image array where voxel intensities represent the Apparent Diffusion Coefficient (ADC). Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.124.3.1.2 Default

- `"first"`

3.124.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.124.3.2 ReferenceImageSelection

3.124.3.2.1 Description The 3D image arrays where each 3D volume corresponds to a single b-value. Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.124.3.2.2 Default

- `"!first"`

3.124.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.124.3.3 NormalizedROILabelRegex

3.124.3.3.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.124.3.3.2 Default

- `".*"`

3.124.3.3.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.124.3.4 ROILabelRegex

3.124.3.4.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.124.3.4.2 Default

- `".*"`

3.124.3.4.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.124.3.5 Model

3.124.3.5.1 Description The model that will be fitted.. Currently, 'adc-simple' , 'adc-ls' , 'biexp', and 'kurtosis' are available. The 'adc-simple' model does not take into account perfusion, only free diffusion is modeled. It only uses the minimum and maximum b-value images and analytically estimates ADC. The 'adc-ls' model, like 'adc-simple', only models free diffusion. It fits a linear least-squares models that uses all available b-value images. The 'kurtosis' model returns 5 parameters corresponding to a biexponential diffsion model with a Kurtosis adjustment, as well as a noise floor parameter added in quadrature with the model. The 'biexp' model uses a segmented fitting approach along with Marquardts method to fit a biexponential decay model, obtaining the pseudodiffusion fraction, pseudodiffusion coefficient, and diffusion coefficient for each voxel.

3.124.3.5.2 Default

- `"adc-simple"`

3.124.3.5.3 Supported Options

- "adc-simple"
- "adc-ls"
- "kurtosis"
- "biexp"

3.124.3.6 Channel

3.124.3.6.1 Description The channel to compare (zero-based). Setting to -1 will compare each channel separately. Note that both test images and reference images must share this specifier.

3.124.3.6.2 Default

- "0"

3.124.3.6.3 Examples

- "-1"
- "0"
- "1"
- "2"

3.124.3.7 TestImgLowerThreshold

3.124.3.7.1 Description Pixel lower threshold for the test images. Only voxels with values above this threshold (inclusive) will be altered.

3.124.3.7.2 Default

- "-inf"

3.124.3.7.3 Examples

- "-inf"
- "0.0"
- "200"

3.124.3.8 TestImgUpperThreshold

3.124.3.8.1 Description Pixel upper threshold for the test images. Only voxels with values below this threshold (inclusive) will be altered.

3.124.3.8.2 Default

- "inf"

3.124.3.8.3 Examples

- `"inf"`
- `"1.23"`
- `"1000"`

3.125 ModifyContourMetadata

3.125.1 Description

This operation injects metadata into contours.

3.125.2 Parameters

- NormalizedROILabelRegex
- ROILabelRegex
- KeyValues

3.125.2.1 NormalizedROILabelRegex

3.125.2.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.125.2.1.2 Default

- `".*"`

3.125.2.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.125.2.2 ROILabelRegex

3.125.2.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.125.2.2.2 Default

- `".*"`

3.125.2.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.125.2.3 KeyValues

3.125.2.3.1 Description Key-value pairs in the form of 'key1@value1;key2@value2' that will be injected into the selected images. Existing metadata will be overwritten. Both keys and values are case-sensitive. Note that a semi-colon separates key-value pairs, not a colon. Note that quotation marks are not stripped internally, but may have to be provided for the shell to properly interpret the argument.

3.125.2.3.2 Default

- `""`

3.125.2.3.3 Examples

- `"Description@'some description'"`
- `"'Description@some description'"`
- `"MinimumSeparation@1.23"`
- `"'Description@some description;MinimumSeparation@1.23'"`

3.126 ModifyImageMetadata

3.126.1 Description

This operation injects metadata into images. It can also modify image spatial characteristics, which are distinct from metadata.

3.126.2 Parameters

- ImageSelection
- KeyValues
- SliceThickness
- VoxelWidth
- VoxelHeight
- ImageAnchor
- ImagePosition
- ImageOrientationColumn
- ImageOrientationRow

3.126.2.1 ImageSelection

3.126.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.126.2.1.2 Default

- `"last"`

3.126.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.126.2.2 KeyValues

3.126.2.2.1 Description Key-value pairs in the form of 'key1@value1;key2@value2' that will be injected into the selected images. Existing metadata will be overwritten. Both keys and values are case-sensitive. Note that a semi-colon separates key-value pairs, not a colon. Note that quotation marks are not stripped internally, but may have to be provided for the shell to properly interpret the argument. Also note that updating spatial metadata will not result in the image characteristics being altered – use the specific parameters provided to update spatial characteristics.

3.126.2.2.2 Default

- `""`

3.126.2.2.3 Examples

- `"Description@'some description'"`
- `"'Description@some description'"`
- `"MinimumSeparation@1.23"`
- `"'Description@some description;MinimumSeparation@1.23'"`

3.126.2.3 SliceThickness

3.126.2.3.1 Description Image slices will be have this thickness (in DICOM units: mm). For most purposes, SliceThickness should be equal to SpacingBetweenSlices. If SpacingBetweenSlices is smaller than SliceThickness, images will overlap. If SpacingBetweenSlices is larger than SliceThickness, there will be a

gap between images. Updating the SliceThickness or image positioning using this operation will alter the image, but will not update SpacingBetweenSlices. This gives the user freedom to alter all image planes individually, allowing construction of non-rectilinear image volumes. If SpacingBetweenSlices is known and consistent, it should be reflected in the image metadata (by the user).

3.126.2.3.2 Default

- `"1.0"`

3.126.2.3.3 Examples

- `"0.1"`
- `"0.5"`
- `"1.0"`
- `"10.0"`

3.126.2.4 VoxelWidth

3.126.2.4.1 Description Voxels will have this (in-plane) width (in DICOM units: mm). This means that row-adjacent voxels centres will be separated by VoxelWidth). Each voxel will have dimensions: VoxelWidth x VoxelHeight x SliceThickness.

3.126.2.4.2 Default

- `"1.0"`

3.126.2.4.3 Examples

- `"0.1"`
- `"0.5"`
- `"1.0"`
- `"10.0"`

3.126.2.5 VoxelHeight

3.126.2.5.1 Description Voxels will have this (in-plane) height (in DICOM units: mm). This means that column-adjacent voxels centres will be separated by VoxelHeight). Each voxel will have dimensions: VoxelWidth x VoxelHeight x SliceThickness.

3.126.2.5.2 Default

- `"1.0"`

3.126.2.5.3 Examples

- "0.1"
- "0.5"
- "1.0"
- "10.0"

3.126.2.6 ImageAnchor

3.126.2.6.1 Description A point in 3D space which denotes the origin (in DICOM units: mm). All other vectors are taken to be relative to this point. Under most circumstance the anchor should be (0,0,0). Specify coordinates separated by commas.

3.126.2.6.2 Default

- "0.0, 0.0, 0.0"

3.126.2.6.3 Examples

- "0.0, 0.0, 0.0"
- "0.0,0.0,0.0"
- "1.0, -2.3, 45.6"

3.126.2.7 ImagePosition

3.126.2.7.1 Description The centre of the row=0, column=0 voxel in the first image (in DICOM units: mm). Specify coordinates separated by commas.

3.126.2.7.2 Default

- "0.0, 0.0, 0.0"

3.126.2.7.3 Examples

- "0.0, 0.0, 0.0"
- "100.0,100.0,100.0"
- "1.0, -2.3, 45.6"

3.126.2.8 ImageOrientationColumn

3.126.2.8.1 Description The orientation unit vector that is aligned with image columns. Care should be taken to ensure ImageOrientationRow and ImageOrientationColumn are orthogonal. (A Gram-Schmidt orthogonalization procedure ensures they are, but the image orientation may not match the expected orientation.) Note that the magnitude will also be scaled to unit length for convenience. Specify coordinates separated by commas.

3.126.2.8.2 Default

- `"1.0, 0.0, 0.0"`

3.126.2.8.3 Examples

- `"1.0, 0.0, 0.0"`
- `"1.0, 1.0, 0.0"`
- `"0.0, 0.0, -1.0"`

3.126.2.9 ImageOrientationRow

3.126.2.9.1 Description The orientation unit vector that is aligned with image rows. Care should be taken to ensure ImageOrientationRow and ImageOrientationColumn are orthogonal. (A Gram-Schmidt orthogonalization procedure ensures they are, but the image orientation may not match the expected orientation.) Note that the magnitude will also be scaled to unit length for convenience. Specify coordinates separated by commas.

3.126.2.9.2 Default

- `"0.0, 1.0, 0.0"`

3.126.2.9.3 Examples

- `"0.0, 1.0, 0.0"`
- `"0.0, 1.0, 1.0"`
- `"-1.0, 0.0, 0.0"`

3.127 NegatePixels

3.127.1 Description

This operation negates pixels for the selected image arrays. This functionality is often desired for processing MR images.

3.127.2 Parameters

- ImageSelection

3.127.2.1 ImageSelection

3.127.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.127.2.1.2 Default

- `"last"`

3.127.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.128 NoOp

3.128.1 Description

This operation does nothing. It produces no side-effects.

3.128.2 Parameters

No registered options.

3.129 NormalizeLineSamples

3.129.1 Description

This operation scales line samples according to a user-provided normalization criteria.

3.129.2 Notes

- Each line sample is independently normalized.

3.129.3 Parameters

- LineSelection
- Method

3.129.3.1 LineSelection

3.129.3.1.1 Description Select one or more line samples. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth line sample (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last line sample. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the line sample composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all line sample that do not have the greatest number of sub-objects, not the least-numerous line sample (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.129.3.1.2 Default

- `"last"`

3.129.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.129.3.2 Method

3.129.3.2.1 Description The type of normalization to apply. The currently supported options are 'area' and 'peak'. 'Area' ensures that the total integrated area is equal to one by scaling the ordinate. 'Peak' ensures that the maximum ordinate is one and the minimum ordinate is zero.

3.129.3.2.2 Default

- `"area"`

3.129.3.2.3 Supported Options

- `"area"`
- `"peak"`

3.130 NormalizePixels

3.130.1 Description

This routine normalizes voxel intensities by adjusting them so they satisfy a 'normalization' criteria. This operation is useful as a pre-processing step when performing convolution or thresholding with absolute magnitudes.

3.130.2 Notes

- This operation considers entire image arrays, not just single images.
- This operation does not *reduce* voxels (i.e., the neighbourhood surrounding is voxel is ignored). This operation effectively applies a linear mapping to every scalar voxel value independently. Neighbourhood-based reductions are implemented in another operation.

3.130.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Inclusivity
- ContourOverlap
- Channel
- Method

3.130.3.1 ImageSelection

3.130.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.130.3.1.2 Default

- "last"

3.130.3.1.3 Examples

- "last"
- "first"
- "all"
- "none"
- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.130.3.2 NormalizedROILabelRegex

3.130.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.130.3.2.2 Default

- ".*"

3.130.3.2.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.130.3.3 ROILabelRegex

3.130.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.130.3.3.2 Default

- `".*"`

3.130.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.130.3.4 Inclusivity

3.130.3.4.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.130.3.4.2 Default

- `"center"`

3.130.3.4.3 Supported Options

- `"center"`
- `"centre"`
- `"planar_corner_inclusive"`
- `"planar_inc"`
- `"planar_corner_exclusive"`
- `"planar_exc"`

3.130.3.5 ContourOverlap

3.130.3.5.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.130.3.5.2 Default

- `"ignore"`

3.130.3.5.3 Supported Options

- `"ignore"`
- `"honour_opposite_orientations"`
- `"overlapping_contours_cancel"`
- `"honour_opps"`
- `"overlap_cancel"`

3.130.3.6 Channel

3.130.3.6.1 Description The channel to operate on (zero-based). Negative values will cause all channels to be operated on.

3.130.3.6.2 Default

- `"0"`

3.130.3.6.3 Examples

- `"-1"`
- `"0"`
- `"1"`

3.130.3.7 Method

3.130.3.7.1 Description Controls the specific type of normalization that will be applied. 'Stretch01' will rescale the voxel values so the minima are 0 and the maxima are 1. Likewise, 'stretch11' will rescale such that the minima are -1 and the maxima are 1. Clamp will ensure all voxel intensities are within [0:1] by setting those lower than 0 to 0 and those higher than 1 to 1. (Voxels already within [0:1] will not be altered.) 'Sum-to-zero' will shift all voxels so that the sum of all voxel intensities is zero. (This is useful for convolution kernels.)

3.130.3.7.2 Default

- `"stretch11"`

3.130.3.7.3 Supported Options

- `"clamp"`
- `"stretch01"`
- `"stretch11"`
- `"sum-to-zero"`

3.131 OptimizeStaticBeams

3.131.1 Description

This operation takes dose matrices corresponding to single, static RT beams and attempts to optimize beam weighting to create an optimal plan subject to various criteria.

3.131.2 Notes

- This routine is a simplisitic routine that attempts to estimate the optimal beam weighting. It should NOT be used for clinical purposes, except maybe as a secondary check or a means to guess reasonable beam weights prior to optimization within the clinical TPS.
- Because beam weights are (generally) not specified in DICOM RTDOSE files, the beam weights are assumed to all be 1.0. If they are not all 1.0, the weights reported here will be relative to whatever the existing weights are.
- If no PTV ROI is available, the BODY contour may suffice. If this is not available, dose outside the body should somehow be set to zero to avoid confusing D_{max} metrics. For example, bolus D_{max} can be high, but is ultimately irrelevant.
- By default, this routine uses all available images. This may be fixed in a future release. Patches are welcome.

3.131.3 Parameters

- ImageSelection
- ResultsSummaryFileName
- UserComment
- NormalizedROILabelRegex
- ROILabelRegex
- MaxVoxelSamples
- NormalizationD

- NormalizationV
- RxDose

3.131.3.1 ImageSelection

3.131.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.131.3.1.2 Default

- `"all"`

3.131.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.131.3.2 ResultsSummaryFileName

3.131.3.2.1 Description This file will contain a brief summary of the results. The format is CSV. Leave empty to dump to generate a unique temporary file. If an existing file is present, rows will be appended without writing a header.

3.131.3.2.2 Default

- `""`

3.131.3.2.3 Examples

- `""`
- `"/tmp/somefile"`
- `"localfile.csv"`
- `"derivative_data.csv"`

3.131.3.3 UserComment

3.131.3.3.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data.

3.131.3.3.2 Default

- `""`

3.131.3.3.3 Examples

- `""`
- `"Using XYZ"`
- `"Patient treatment plan C"`

3.131.3.4 NormalizedROILabelRegex

3.131.3.4.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful

for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.131.3.4.2 Default

- `".*"`

3.131.3.4.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.131.3.5 ROILabelRegex

3.131.3.5.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.131.3.5.2 Default

- `".*"`

3.131.3.5.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.131.3.6 MaxVoxelSamples

3.131.3.6.1 Description The maximum number of voxels to randomly sample (deterministically) within the PTV. Setting lower will result in faster calculation, but lower precision. A reasonable setting depends on the size of the target structure; small targets may suffice with a few hundred voxels, but larger targets probably require several thousand.

3.131.3.6.2 Default

- `"1000"`

3.131.3.6.3 Examples

- `"200"`
- `"500"`
- `"1000"`
- `"2000"`
- `"5000"`

3.131.3.7 NormalizationD

3.131.3.7.1 Description The isodose value that should envelop a given volume in the PTV ROI. In other words, this parameter is the 'D' parameter in a DVH constraint of the form $V_D \geq V_{min}$. It should be given as a fraction within [0:1] relative to the prescription dose. For example, 95% isodose should be provided as '0.95'.

3.131.3.7.2 Default

- `"0.95"`

3.131.3.7.3 Examples

- `"0.90"`
- `"0.95"`
- `"0.98"`
- `"0.99"`
- `"1.0"`

3.131.3.8 NormalizationV

3.131.3.8.1 Description The minimal fractional volume of ROI that should be enclosed within one or more surfaces that demarcate the given isodose value. In other words, this parameter is the 'Vmin' parameter in a DVH constraint of the form $V_D \geq V_{min}$. It should be given as a fraction within [0:1] relative to the volume of the ROI (typically discretized to the number of voxels in the ROI). For example, if Vmin = 99%, provide the value '0.99'.

3.131.3.8.2 Default

- `"0.99"`

3.131.3.8.3 Examples

- `"0.90"`
- `"0.95"`
- `"0.98"`
- `"0.99"`
- `"1.0"`

3.131.3.9 RxDose

3.131.3.9.1 Description The dose prescribed to the ROI that will be optimized. The units depend on the DICOM file, but will likely be Gy.

3.131.3.9.2 Default

- `"70.0"`

3.131.3.9.3 Examples

- `"48.0"`
- `"60.0"`
- `"63.3"`
- `"70.0"`
- `"100.0"`

3.132 OrderImages

3.132.1 Description

This operation will order individual image slices within collections (Image_Arrays) based on the values of the specified metadata tags.

3.132.2 Notes

- Images are moved, not copied.
- Image groupings are retained, and the order of groupings is not altered.
- Images that do not contain the specified metadata will be sorted after the end.

3.132.3 Parameters

- ImageSelection
- Key

3.132.3.1 ImageSelection

3.132.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.132.3.1.2 Default

- `"all"`

3.132.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`

- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.132.3.2 Key

3.132.3.2.1 Description Image metadata key to use for ordering. Images will be sorted according to the key's value 'natural' sorting order, which compares sub-strings of numbers and characters separately. Note this ordering is expected to be stable, but may not always be on some systems.

3.132.3.2.2 Default

- `""`

3.132.3.2.3 Examples

- `"AcquisitionTime"`
- `"ContentTime"`
- `"SeriesNumber"`
- `"SeriesDescription"`

3.133 PartitionContours

3.133.1 Description

This operation partitions the selected contours, producing a number of subsegments that could be re-combined to re-create the original contours.

3.133.2 Parameters

- ROILabelRegex
- NormalizedROILabelRegex
- PlanarOrientation
- SubsegmentRootROILabel
- SubsegMethod
- NestedCleaveOrder
- XPartitions
- YPartitions
- ZPartitions
- ReverseXTraversalOrder
- ReverseYTraversalOrder
- ReverseZTraversalOrder
- FractionalTolerance
- MaxBisects

3.133.2.1 ROILabelRegex

3.133.2.1.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.133.2.1.2 Default

- `".*"`

3.133.2.1.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.133.2.2 NormalizedROILabelRegex

3.133.2.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.133.2.2.2 Default

- `".*"`

3.133.2.2.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.133.2.3 PlanarOrientation

3.133.2.3.1 Description A string instructing how to orient the cleaving planes. Currently supported: (1) 'axis-aligned' (i.e., align with the image/dose grid row and column unit vectors) and (2) 'static-oblique' (i.e., same as axis-aligned but rotated 22.5 degrees to reduce colinearity, which sometimes improves sub-segment area consistency).

3.133.2.3.2 Default

- "axis-aligned"

3.133.2.3.3 Supported Options

- "axis-aligned"
- "static-oblique"

3.133.2.4 SubsegmentRootROILabel

3.133.2.4.1 Description The root ROI label to attach to the sub-segments. The full name will be this root followed by '_' and the number of the subsegment.

3.133.2.4.2 Default

- "subsegment"

3.133.2.4.3 Examples

- "subsegment"
- "ss"
- "partition"

3.133.2.5 SubsegMethod

3.133.2.5.1 Description The method to use for sub-segmentation. Nested sub-segmentation should almost always be preferred unless you know what you're doing. It should be faster too. Compound sub-segmentation is known to cause problems, e.g., with zero-area sub-segments and spatial dependence in sub-segment volume. Nested cleaving will produce sub-segments of equivalent area (volume) throughout the entire ROI whereas compound sub-segmentation will not.

3.133.2.5.2 Default

- `"nested-cleave"`

3.133.2.5.3 Supported Options

- `"nested-cleave"`
- `"compound-cleave"`

3.133.2.6 NestedCleaveOrder

3.133.2.6.1 Description The order in which to apply nested cleaves. This routine requires one of 'ZYX', 'ZXY', 'XYZ', 'XZY', 'YZX', or 'YXZ'. Cleaves are implemented from left to right using the specified X, Y, and Z selection criteria.

3.133.2.6.2 Default

- `"ZXY"`

3.133.2.6.3 Supported Options

- `"ZXY"`
- `"ZYX"`
- `"XYZ"`
- `"XZY"`
- `"YZX"`
- `"YXZ"`

3.133.2.7 XPartitions

3.133.2.7.1 Description The number of partitions to find along the 'X' axis. The total number of sub-segments produced along the 'X' axis will be (1+XPartitions). A value of zero will disable the partitioning along the 'X' axis.

3.133.2.7.2 Default

- `"0"`

3.133.2.7.3 Examples

- "0"
- "1"
- "3"
- "5"
- "50"

3.133.2.8 YPartitions

3.133.2.8.1 Description The number of partitions to find along the 'Y' axis. The total number of sub-segments produced along the 'Y' axis will be (1+YPartitions). A value of zero will disable the partitioning along the 'Y' axis.

3.133.2.8.2 Default

- "0"

3.133.2.8.3 Examples

- "0"
- "1"
- "3"
- "5"
- "50"

3.133.2.9 ZPartitions

3.133.2.9.1 Description The number of partitions to find along the 'Z' axis. The total number of sub-segments produced along the 'Z' axis will be (1+ZPartitions). A value of zero will disable the partitioning along the 'Z' axis.

3.133.2.9.2 Default

- "0"

3.133.2.9.3 Examples

- "0"
- "1"
- "3"
- "5"
- "50"

3.133.2.10 ReverseXTraversalOrder

3.133.2.10.1 Description Controls the order in which sub-segments are numbered. If set to 'true' the numbering will be reversed along the X axis. This option is most useful when the 'X' axis intersects mirrored ROIs (e.g., left and right parotid glands).

3.133.2.10.2 Default

- `"false"`

3.133.2.10.3 Examples

- `"false"`
- `"true"`

3.133.2.11 ReverseYTraversalOrder

3.133.2.11.1 Description Controls the order in which sub-segments are numbered. If set to 'true' the numbering will be reversed along the Y axis. This option is most useful when the 'Y' axis intersects mirrored ROIs (e.g., left and right parotid glands).

3.133.2.11.2 Default

- `"false"`

3.133.2.11.3 Examples

- `"false"`
- `"true"`

3.133.2.12 ReverseZTraversalOrder

3.133.2.12.1 Description Controls the order in which sub-segments are numbered. If set to 'true' the numbering will be reversed along the Z axis. This option is most useful when the 'Z' axis intersects mirrored ROIs (e.g., left and right parotid glands).

3.133.2.12.2 Default

- `"false"`

3.133.2.12.3 Examples

- `"false"`
- `"true"`

3.133.2.13 FractionalTolerance

3.133.2.13.1 Description The tolerance of X, Y, and Z fractional area bisection criteria (see ZSelection description). This parameter specifies a stopping condition for the bisection procedure. If it is set too high, sub-segments may be inadequatly rough. If it is set too low, bisection below the machine precision floor may be attempted, which will result in instabilities. Note that the number of permitted iterations will control whether this tolerance can possibly be reached; if strict adherence is required, set the maximum number of iterations to be excessively large.

3.133.2.13.2 Default

- `"0.001"`

3.133.2.13.3 Examples

- `"1E-2"`
- `"1E-3"`
- `"1E-4"`
- `"1E-5"`

3.133.2.14 MaxBisects

3.133.2.14.1 Description The maximum number of iterations the bisection procedure can perform. This parameter specifies a stopping condition for the bisection procedure. If it is set too low, sub-segments may be inadequatly rough. If it is set too high, bisection below the machine precision floor may be attempted, which will result in instabilities. Note that the fractional tolerance will control whether this tolerance can possibly be reached; if an exact number of iterations is required, set the fractional tolerance to be excessively small.

3.133.2.14.2 Default

- `"20"`

3.133.2.14.3 Examples

- `"10"`
- `"20"`
- `"30"`

3.134 PlotLineSamples

3.134.1 Description

This operation plots the selected line samples.

3.134.2 Parameters

- LineSelection
- Title
- AbscissaLabel
- OrdinateLabel

3.134.2.1 LineSelection

3.134.2.1.1 Description Select one or more line samples. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth line sample (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last line sample. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the line sample composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all line sample that do not have the greatest number of sub-objects, not the least-numerous line sample (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.134.2.1.2 Default

- `"last"`

3.134.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`

- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.134.2.2 Title

3.134.2.2.1 Description The title to display in the plot. Leave empty to disable.

3.134.2.2.2 Default

- `""`

3.134.2.2.3 Examples

- `"Line Samples"`
- `"Time Series"`
- `"DVH for XYZ"`

3.134.2.3 AbscissaLabel

3.134.2.3.1 Description The label to attach to the abscissa (i.e., the 'x' or horizontal coordinate). Leave empty to disable.

3.134.2.3.2 Default

- `""`

3.134.2.3.3 Examples

- `"(arb.)"`
- `"Time (s)"`
- `"Distance (mm)"`
- `"Dose (Gy)"`

3.134.2.4 OrdinateLabel

3.134.2.4.1 Description The label to attach to the ordinate (i.e., the 'y' or vertical coordinate). Leave empty to disable.

3.134.2.4.2 Default

- `""`

3.134.2.4.3 Examples

- `"(arb.)"`
- `"Intensity (arb.)"`
- `"Volume (mm^3)"`
- `"Fraction (arb.)"`

3.135 PlotPerROITimeCourses

3.135.1 Description

Interactively plot time courses for the specified ROI(s).

3.135.2 Parameters

- ROILabelRegex

3.135.2.1 ROILabelRegex

3.135.2.1.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.135.2.1.2 Default

- `".*"`

3.135.2.1.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.136 PointSeparation

3.136.1 Description

This operation estimates the minimum and maximum point-to-point separation between two point clouds. It also computes the longest-nearest (Hausdorff) separation, i.e., the length of the longest lines from points in selection A to the nearest point in selection B.

3.136.2 Notes

- This routine scales like $O(N * M)$ where N and M are the number of points in each point cloud. No indexing or acceleration is used. Beware that large point clouds will result in slow computations.
- This operation can be used to compare points clouds that are nearly alike. Such comparisons are useful for quantifying discrepancies after transformations, reconstructions, simplifications, or any other scenarios where a point cloud must be reasonable accurately reproduced.

3.136.3 Parameters

- PointSelectionA
- PointSelectionB
- FileName
- UserComment

3.136.3.1 PointSelectionA

3.136.3.1.1 Description Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note

that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.136.3.1.2 Default

- `"first"`

3.136.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.136.3.2 PointSelectionB

3.136.3.2.1 Description Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.136.3.2.2 Default

- `"last"`

3.136.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.136.3.3 FileName

3.136.3.3.1 Description A filename (or full path) in which to append separation data generated by this routine. The format is CSV. Leave empty to dump to generate a unique temporary file.

3.136.3.3.2 Default

- `""`

3.136.3.3.3 Examples

- `""`
- `"/tmp/somefile"`
- `"localfile.csv"`
- `"derivative_data.csv"`

3.136.3.4 UserComment

3.136.3.4.1 Description A string that will be inserted into the output file which will simplify merging output with differing parameters, from different sources, or using sub-selections of the data.

3.136.3.4.2 Default

- ""

3.136.3.4.3 Examples

- ""
- "Using XYZ"
- "Patient treatment plan C"

3.137 PreFilterEnormousCTValues

3.137.1 Description

This operation runs the data through a per-pixel filter, censoring pixels which are too high to legitimately show up in a clinical CT. Censored pixels are set to NaN. Data is modified and no copy is made!

3.137.2 Parameters

No registered options.

3.138 PresentationImage

3.138.1 Description

This operation renders an image with any contours in-place and colour mapping using an SFML backend.

3.138.2 Notes

- By default this operation displays the last available image. This makes it easier to produce a sequence of images by inserting this operation into a sequence of operations.

3.138.3 Parameters

- ScaleFactor
- ImageFileName
- ColourMapRegex
- WindowLow
- WindowHigh

3.138.3.1 ScaleFactor

3.138.3.1.1 Description This factor is applied to the image width and height to magnify (larger than 1) or shrink (less than 1) the image. This factor only affects the output image size. Note that aspect ratio is retained, but rounding for non-integer factors may lead to small (1-2 pixel) discrepancies.

3.138.3.1.2 Default

- `"1.0"`

3.138.3.1.3 Examples

- `"0.5"`
- `"1.0"`
- `"2.0"`
- `"5.23"`

3.138.3.2 ImageFileName

3.138.3.2.1 Description The file name to use for the image. If blank, a filename will be generated sequentially.

3.138.3.2.2 Default

- `""`

3.138.3.2.3 Examples

- `""`
- `"/tmp/an_image.png"`
- `"afile.png"`

3.138.3.3 ColourMapRegex

3.138.3.3.1 Description The colour mapping to apply to the image if there is a single channel. The default will match the first available, and if there is no matching map found, the first available will be selected.

3.138.3.3.2 Default

- `".*"`

3.138.3.3.3 Supported Options

- `"Viridis"`
- `"Magma"`
- `"Plasma"`
- `"Inferno"`

- "Jet"
- "MorelandBlueRed"
- "MorelandBlackBody"
- "MorelandExtendedBlackBody"
- "KRC"
- "ExtendedKRC"
- "Kovesi_LinKRYW_5-100_c64"
- "Kovesi_LinKRYW_0-100_c71"
- "Kovesi_Cyclic_cet-c2"
- "LANLOliveGreentoBlue"
- "YgorIncandescent"
- "LinearRamp"

3.138.3.4 WindowLow

3.138.3.4.1 Description If provided, this parameter will override any existing window and level. All pixels with the intensity value or lower will be assigned the lowest possible colour according to the colour map. Not providing a valid number will disable window overrides.

3.138.3.4.2 Default

- ""

3.138.3.4.3 Examples

- ""
- "-1.23"
- "0"
- "1E4"

3.138.3.5 WindowHigh

3.138.3.5.1 Description If provided, this parameter will override any existing window and level. All pixels with the intensity value or higher will be assigned the highest possible colour according to the colour map. Not providing a valid number will disable window overrides.

3.138.3.5.2 Default

- ""

3.138.3.5.3 Examples

- ""
- "1.23"

- "0"
- "10.3E4"

3.139 PruneEmptyImageDoseArrays

3.139.1 Description

This operation deletes Image_Arrays that do not contain any images.

3.139.2 Parameters

No registered options.

3.140 PurgeContours

3.140.1 Description

This routine purges (deletes) individual contours if they satisfy various criteria.

3.140.2 Notes

- This operation considers only individual contours at the moment. It could be extended to operate on whole ROIs (i.e., contour_collections), or to perform a separate vote within each ROI. The individual contour approach was taken since filtering out small contour 'islands' is the primary use-case.

3.140.3 Parameters

- ROILabelRegex
- NormalizedROILabelRegex
- InvertLogic
- InvertSelection
- AreaAbove
- AreaBelow
- PerimeterAbove
- PerimeterBelow
- VertexCountAbove
- VertexCountBelow

3.140.3.1 ROILabelRegex

3.140.3.1.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.140.3.1.2 Default

- `".*"`

3.140.3.1.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.140.3.2 NormalizedROILabelRegex

3.140.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.140.3.2.2 Default

- `".*"`

3.140.3.2.3 Examples

- `".*"`
- `".*Body.*"`

- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.140.3.3 InvertLogic

3.140.3.3.1 Description This option controls whether purged contours *should* or *should not* satisfy the specified logical test criteria. If false (the default), this operation is equivalent to a 'purge if and only if' operation. If true, this operation is equivalent to a 'retain if and only if' operation. Note that this parameter is independent of the ROI selection criteria.

3.140.3.3.2 Default

- "false"

3.140.3.3.3 Examples

- "true"
- "false"

3.140.3.4 InvertSelection

3.140.3.4.1 Description This option controls whether purged contours *should* or *should not* satisfy the specified ROI selection criteria. If false (the default), this operation only considers contours that match the ROILabelRegex or NormalizedROILabelRegex; all other contours are ignored (and thus will not be purged, i.e., a denylist). If true, this operation only considers the *complement* of contours that match the ROILabelRegex or NormalizedROILabelRegex, which can be used to purge all contours except a handful (i.e., an allowlist).

3.140.3.4.2 Default

- "false"

3.140.3.4.3 Examples

- "true"
- "false"

3.140.3.5 AreaAbove

3.140.3.5.1 Description If this option is provided with a valid positive number, contour(s) with an area greater than the specified value are purged. Note that the DICOM coordinate space is used. (Supplying the default, inf, will effectively disable this option.)

3.140.3.5.2 Default

- `"inf"`

3.140.3.5.3 Examples

- `"inf"`
- `"100.0"`
- `"1000"`
- `"10.23E8"`

3.140.3.6 AreaBelow

3.140.3.6.1 Description If this option is provided with a valid positive number, contour(s) with an area less than the specified value are purged. Note that the DICOM coordinate space is used. (Supplying the default, -inf, will effectively disable this option.)

3.140.3.6.2 Default

- `"-inf"`

3.140.3.6.3 Examples

- `"-inf"`
- `"100.0"`
- `"1000"`
- `"10.23E8"`

3.140.3.7 PerimeterAbove

3.140.3.7.1 Description If this option is provided with a valid positive number, contour(s) with a perimeter greater than the specified value are purged. Note that the DICOM coordinate space is used. (Supplying the default, inf, will effectively disable this option.)

3.140.3.7.2 Default

- `"inf"`

3.140.3.7.3 Examples

- `"inf"`
- `"10.0"`
- `"100"`
- `"10.23E4"`

3.140.3.8 PerimeterBelow

3.140.3.8.1 Description If this option is provided with a valid positive number, contour(s) with a perimeter less than the specified value are purged. Note that the DICOM coordinate space is used. (Supplying the default, -inf, will effectively disable this option.)

3.140.3.8.2 Default

- `"-inf"`

3.140.3.8.3 Examples

- `"-inf"`
- `"10.0"`
- `"100"`
- `"10.23E4"`

3.140.3.9 VertexCountAbove

3.140.3.9.1 Description If this option is provided with a valid positive number, contour(s) with a vertex count greater than the specified value are purged. Note that the DICOM coordinate space is used. (Supplying the default, inf, will effectively disable this option.)

3.140.3.9.2 Default

- `"inf"`

3.140.3.9.3 Examples

- `"inf"`
- `"10.0"`
- `"100"`
- `"10.23E4"`

3.140.3.10 VertexCountBelow

3.140.3.10.1 Description If this option is provided with a valid positive number, contour(s) with a vertex count less than the specified value are purged. Note that the DICOM coordinate space is used. (Supplying the default, -inf, will effectively disable this option.)

3.140.3.10.2 Default

- `"-inf"`

3.140.3.10.3 Examples

- `"-inf"`
- `"10.0"`
- `"100"`
- `"10.23E4"`

3.141 RankPixels

3.141.1 Description

This operation ranks pixels throughout an image array.

3.141.2 Notes

- This routine operates on all images in an image array, so pixel value ranks are valid throughout the array. However, the window and level of each window is separately determined. You will need to set a uniform window and level manually if desired.
- This routine operates on all images in an image array. If images need to be processed individually, arrays will have to be exploded prior to calling this routine. Note that if this functionality is required, it can be implemented as an operation option easily. Likewise, if multiple image arrays must be considered simultaneously, they will need to be combined before invoking this operation.

3.141.3 Parameters

- ImageSelection
- Method
- LowerThreshold
- UpperThreshold

3.141.3.1 ImageSelection

3.141.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.141.3.1.2 Default

- `"last"`

3.141.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.141.3.2 Method

3.141.3.2.1 Description Pixels participating in the ranking process will have their pixel values replaced. They can be replaced with either a rank or the corresponding percentile. Ranks start at zero and percentiles are centre-weighted (i.e., rank-averaged).

3.141.3.2.2 Default

- `"Percentile"`

3.141.3.2.3 Supported Options

- `"Rank"`
- `"Percentile"`

3.141.3.3 LowerThreshold

3.141.3.3.1 Description The (inclusive) threshold above which pixel values must be in order to participate in the rank.

3.141.3.3.2 Default

- `"-inf"`

3.141.3.3.3 Examples

- `"-inf"`
- `"0.0"`
- `"-900"`

3.141.3.4 UpperThreshold

3.141.3.4.1 Description The (inclusive) threshold below which pixel values must be in order to participate in the rank.

3.141.3.4.2 Default

- `"inf"`

3.141.3.4.3 Examples

- `"inf"`
- `"0.0"`
- `"1500"`

3.142 ReduceNeighbourhood

3.142.1 Description

This routine walks the voxels of a 3D rectilinear image collection, reducing the distribution of voxels within the local volumetric neighbourhood to a scalar value, and updating the voxel value with this scalar. This routine can be used to implement mean and median filters (amongst others) that operate over a variety of 3D neighbourhoods. Besides purely statistical reductions, logical reductions can be applied.

3.142.2 Notes

- The provided image collection must be rectilinear.

- This operation can be used to compute core 3D morphology operations (erosion and dilation) as well as composite operations like opening (i.e., erosion followed by dilation), closing (i.e., dilation followed by erosion), 'gradient' (i.e., the difference between dilation and erosion, which produces an outline), and various other combinations of core and composite operations.

3.142.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Channel
- Neighbourhood
- Reduction
- MaxDistance

3.142.3.1 ImageSelection

3.142.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.142.3.1.2 Default

- `"last"`

3.142.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.142.3.2 NormalizedROILabelRegex

3.142.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.142.3.2.2 Default

- `".*"`

3.142.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.142.3.3 ROILabelRegex

3.142.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.142.3.3.2 Default

- `".*"`

3.142.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.142.3.4 Channel

3.142.3.4.1 Description The channel to operated on (zero-based). Negative values will cause all channels to be operated on.

3.142.3.4.2 Default

- `"0"`

3.142.3.4.3 Examples

- `"-1"`
- `"0"`
- `"1"`

3.142.3.5 Neighbourhood

3.142.3.5.1 Description Controls how the neighbourhood surrounding a voxel is defined. Variable-size neighbourhoods 'spherical' and 'cubic' are defined. An appropriate isotropic extent must be provided for these neighbourhoods. (See below; extents must be provided in DICOM units, i.e., mm.) Fixed-size

neighbourhoods specify a fixed number of adjacent voxels. Fixed rectagular neighbourhoods are specified like 'RxCxI' for row, column, and image slice extents (as integer number of rows, columns, and slices). Fixed spherical neighbourhoods are specified like 'Wsphere' where W is the width (i.e., the number of voxels wide). In morphological terminology, the neighbourhood is referred to as a 'structuring element.' A similar concept is the convolutional 'kernel.'

3.142.3.5.2 Default

- `"spherical"`

3.142.3.5.3 Examples

- `"spherical"`
- `"cubic"`
- `"3x3x3"`
- `"5x5x5"`
- `"3sphere"`
- `"5sphere"`
- `"7sphere"`
- `"9sphere"`
- `"11sphere"`
- `"13sphere"`
- `"15sphere"`

3.142.3.6 Reduction

3.142.3.6.1 Description Controls how the distribution of voxel values from neighbouring voxels is reduced. Statistical distribution reducers 'min', 'mean', 'median', and 'max' are defined. 'min' is also known as the 'erosion' operation. Likewise, 'max' is also known as the 'dilation' operation. Note that the morphological 'opening' operation can be accomplished by sequentially performing an erosion and then a dilation using the same neighbourhood. The 'standardize' reduction method can be used for adaptive rescaling by subtracting the local neighbourhood mean and dividing the local neighbourhood standard deviation. The 'standardize' reduction method is a way to (locally) transform variables on different scales so they can more easily be compared. Note that standardization can result in undefined voxel values when the local neighbourhood is perfectly uniform. Also, since only the local neighbourhood is considered, voxels will in general have *neither* zero mean *nor* a unit standard deviation (growing the neighbourhood extremely large *will* accomplish this, but the calculation will be inefficient). The 'percentile01' reduction method evaluates which percentile the central voxel occupies within the local neighbourhood. It is reported scaled to $[0, 1]$. 'percentile01' can be used to implement non-parametric adaptive scaling since only the local neighbourhood is examined. (Duplicate values assume the

percentile of the middle of the range.) In contrast to 'standardize', the 'percentile01' reduction should remain valid anywhere the local neighbourhood has a non-zero number of finite voxels. Logical reducers 'is_min' and 'is_max' are also available – is_min (is_max) replace the voxel value with 1.0 if it was the min (max) in the neighbourhood and 0.0 otherwise. Logical reducers 'is_min_nan' and 'is_max_nan' are variants that replace the voxel with a NaN instead of 1.0 and otherwise do not overwrite the original voxel value.

3.142.3.6.2 Default

- `"median"`

3.142.3.6.3 Supported Options

- `"min"`
- `"erode"`
- `"mean"`
- `"median"`
- `"max"`
- `"dilate"`
- `"standardize"`
- `"percentile01"`
- `"is_min"`
- `"is_max"`
- `"is_min_nan"`
- `"is_max_nan"`

3.142.3.7 MaxDistance

3.142.3.7.1 Description The maximum distance (inclusive, in DICOM units: mm) within which neighbouring voxels will be evaluated for variable-size neighbourhoods. Note that this parameter will be ignored if a fixed-size neighbourhood has been specified. For spherical neighbourhoods, this distance refers to the radius. For cubic neighbourhoods, this distance refers to 'box radius' or the distance from the cube centre to the nearest point on each bounding face. Voxels separated by more than this distance will not be evaluated together.

3.142.3.7.2 Default

- `"2.0"`

3.142.3.7.3 Examples

- `"0.5"`
- `"2.0"`
- `"15.0"`

3.143 RemeshSurfaceMeshes

3.143.1 Description

This operation re-meshes existing surface meshes according to the specified criteria, replacing the original meshes with remeshed copies.

3.143.2 Notes

- Selected surface meshes should represent polyhedra.

3.143.3 Parameters

- MeshSelection
- Iterations
- TargetEdgeLength

3.143.3.1 MeshSelection

3.143.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.143.3.1.2 Default

- `"last"`

3.143.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.143.3.2 Iterations

3.143.3.2.1 Description The number of remeshing iterations to perform.

3.143.3.2.2 Default

- `"5"`

3.143.3.2.3 Examples

- `"1"`
- `"3"`
- `"5"`
- `"10"`

3.143.3.3 TargetEdgeLength

3.143.3.3.1 Description The desired length of all edges in the remeshed mesh in DICOM units (mm).

3.143.3.3.2 Default

- `"1.5"`

3.143.3.3.3 Examples

- `"0.2"`
- `"0.75"`
- `"1.0"`

- `"1.5"`
- `"2.015"`

3.144 Repeat

3.144.1 Description

This operation is a control flow meta-operation that repeatedly and sequentially invokes all child operations the given number of times.

3.144.2 Notes

- If this operation has no children, this operation will evaluate to a no-op.
- Each repeat is performed sequentially, and all side-effects are carried forward for each iteration. In particular, all selectors in child operations are evaluated lazily, at the moment when the child operation is invoked.
- This operation will most often be used to repeat operations that compose naturally, such as repeatedly applying a small Gaussian filter to simulate a single Gaussian filter with a large kernel, iteratively refining a calculation, loading multiple copies of the same file, or attempting a given analysis while waiting for data from a remote server.

3.144.3 Parameters

- N

3.144.3.1 N

3.144.3.1.1 Description The number of times to repeat the children operations.

3.144.3.1.2 Default

- `"0"`

3.144.3.1.3 Examples

- `"0"`
- `"1"`
- `"5"`
- `"10"`
- `"1000"`

3.145 SDL_Viewer

3.145.1 Description

Launch an interactive viewer based on SDL.

3.145.2 Parameters

No registered options.

3.146 SFML_Viewer

3.146.1 Description

Launch an interactive viewer based on SFML. Using this viewer, it is possible to contour ROIs, generate plots of pixel intensity along profiles or through time, inspect and compare metadata, and various other things.

3.146.2 Parameters

- SingleScreenshot
- SingleScreenshotFileName
- FPSLimit

3.146.2.1 SingleScreenshot

3.146.2.1.1 Description If 'true', a single screenshot is taken and then the viewer is exited. This option works best for quick visual inspections, and should not be used for later processing or analysis.

3.146.2.1.2 Default

- `"false"`

3.146.2.1.3 Examples

- `"true"`
- `"false"`

3.146.2.2 SingleScreenshotFileName

3.146.2.2.1 Description Iff invoking the 'SingleScreenshot' argument, use this string as the screenshot filename. If blank, a filename will be generated sequentially.

3.146.2.2.2 Default

- ""

3.146.2.2.3 Examples

- ""
- "/tmp/a_screenshot.png"
- "afile.png"

3.146.2.3 FPSLimit

3.146.2.3.1 Description The upper limit on the frame rate, in seconds as an unsigned integer. Note that this value may be treated as a suggestion.

3.146.2.3.2 Default

- "60"

3.146.2.3.3 Examples

- "60"
- "30"
- "10"
- "1"

3.147 ScalePixels

3.147.1 Description

This operation scales pixel (voxel) values confined to one or more ROIs.

3.147.2 Notes

- This routine could be used to derive, for example, per-fraction dose from a total dose image array.

3.147.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Inclusivity
- ContourOverlap
- ScaleFactor
- Channel

3.147.3.1 ImageSelection

3.147.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.147.3.1.2 Default

- `"last"`

3.147.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.147.3.2 NormalizedROILabelRegex

3.147.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.147.3.2.2 Default

- `".*"`

3.147.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.147.3.3 ROILabelRegex

3.147.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.147.3.3.2 Default

- `".*"`

3.147.3.3.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.147.3.4 Inclusivity

3.147.3.4.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.147.3.4.2 Default

- "center"

3.147.3.4.3 Supported Options

- "center"
- "centre"
- "planar_corner_inclusive"
- "planar_inc"
- "planar_corner_exclusive"
- "planar_exc"

3.147.3.5 ContourOverlap

3.147.3.5.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.147.3.5.2 Default

- "ignore"

3.147.3.5.3 Supported Options

- "ignore"
- "honour_opposite_orientations"
- "overlapping_contours_cancel"
- "honour_opps"
- "overlap_cancel"

3.147.3.6 ScaleFactor

3.147.3.6.1 Description The numeric factor to multiply all pixel (voxel) values with.

3.147.3.6.2 Default

- "1.0"

3.147.3.6.3 Examples

- "-1.0"
- "0.0"
- "1.23E-5"

3.147.3.7 Channel

3.147.3.7.1 Description The image channel to use. Zero-based.

3.147.3.7.2 Default

- "0"

3.147.3.7.3 Examples

- "0"
- "1"
- "2"

3.148 SeamContours

3.148.1 Description

This routine converts contours that represent 'outer' and 'inner' via contour orientation into contours that are uniformly outer but have a zero-area seam connecting the inner and outer portions.

3.148.2 Notes

- This routine currently operates on all available ROIs.
- This routine operates on one contour_collection at a time. It will combine contours that are in the same contour_collection and overlap, even if they have different ROINames. Consider making a complementary routine that partitions contours into ROIs based on ROIName (or other metadata) if more rigorous enforcement is needed.
- This routine actually computes the XOR Boolean of contours that overlap. So if contours partially overlap, this routine will treat the overlapping parts as if they are holes, and the non-overlapping parts as if they represent the ROI. This behaviour may be surprising in some cases.
- This routine will also treat overlapping contours with like orientation as if the smaller contour were a hole of the larger contour.
- This routine will ignore contour orientation if there is only a single contour. More specifically, for a given ROI label, planes with a single contour will be unaltered.
- Only the common metadata between outer and inner contours is propagated to the seamed contours.
- This routine will NOT combine disconnected contours with a seam. Disconnected contours will remain disconnected.

3.148.3 Parameters

No registered options.

3.149 SelectSlicesIntersectingROI

3.149.1 Description

This operation applies a whitelist to the most-recently loaded images. Images must 'slice' through one of the described ROIs in order to make the whitelist. This operation is typically used to reduce long computations by trimming the field of view of extraneous image slices.

3.149.2 Parameters

- NormalizedROILabelRegex
- ROILabelRegex

3.149.2.1 NormalizedROILabelRegex

3.149.2.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.149.2.1.2 Default

- `".*"`

3.149.2.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.149.2.2 ROILabelRegex

3.149.2.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.149.2.2.2 Default

- `".*"`

3.149.2.2.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.150 SimplifyContours

3.150.1 Description

This operation performs simplification on contours by removing or moving vertices. This operation is mostly used to reduce the computational complexity of other operations.

3.150.2 Notes

- Contours are currently processed individually, not as a volume.
- Simplification is generally performed most eagerly on regions with relatively low curvature. Regions of high curvature are generally simplified only as necessary.

3.150.3 Parameters

- NormalizedROILabelRegex
- ROILabelRegex
- FractionalAreaTolerance
- SimplificationMethod

3.150.3.1 NormalizedROILabelRegex

3.150.3.1.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful

for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.150.3.1.2 Default

- `".*"`

3.150.3.1.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.150.3.2 ROILabelRegex

3.150.3.2.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.150.3.2.2 Default

- `".*"`

3.150.3.2.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.150.3.3 FractionalAreaTolerance

3.150.3.3.1 Description The fraction of area each contour will tolerate during simplified. This is a measure of how much the contour area can change due to simplification.

3.150.3.3.2 Default

- `"0.01"`

3.150.3.3.3 Examples

- `"0.001"`
- `"0.01"`
- `"0.02"`
- `"0.05"`
- `"0.10"`

3.150.3.4 SimplificationMethod

3.150.3.4.1 Description The specific algorithm used to perform contour simplification. 'Vertex removal' is a simple algorithm that removes vertices one-by-one without replacement. It iteratively ranks vertices and removes the single vertex that has the least impact on contour area. It is best suited to removing redundant vertices or whenever new vertices should not be added. 'Vertex collapse' combines two adjacent vertices into a single vertex at their midpoint. It iteratively ranks vertex pairs and removes the single vertex that has the least total impact on contour area. Note that small sharp features that alternate inward and outward will have a small total area cost, so will be pruned early. Thus this technique acts as a low-pass filter and will defer simplification of high-curvature regions until necessary. It is more economical compared to vertex removal in that it will usually simplify contours more for a given tolerance (or, equivalently, can retain contour fidelity better than vertex removal for the same number of vertices). However, vertex collapse performs an averaging that may result in numerical imprecision.

3.150.3.4.2 Default

- `"vert-collapse"`

3.150.3.4.3 Supported Options

- `"vertex-collapse"`
- `"vertex-removal"`

3.151 SimplifySurfaceMeshes

3.151.1 Description

This operation performs mesh simplification on existing surface meshes according to the specified criteria, replacing the original meshes with simplified copies.

3.151.2 Notes

- Selected surface meshes should represent polyhedra.

3.151.3 Parameters

- MeshSelection
- EdgeCountLimit

3.151.3.1 MeshSelection

3.151.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.151.3.1.2 Default

- `"last"`

3.151.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.151.3.2 EdgeCountLimit

3.151.3.2.1 Description The maximum number of edges simplified meshes should contain.

3.151.3.2.2 Default

- `"250000"`

3.151.3.2.3 Examples

- `"20000"`
- `"100000"`
- `"500000"`
- `"5000000"`

3.152 SimulateRadiograph

3.152.1 Description

This routine uses ray marching and volumetric sampling to simulate radiographs using a CT image array. Voxels are assumed to have intensities in HU. A simplisitic conversion from CT number (in HU) to relative electron density (see note below) is performed for marched rays.

3.152.2 Notes

- Images must be regular.
- This operation currently takes a simplistic approach and should only be used for purposes where the simulated radiograph contrast can be tuned and validated (e.g., in a relative way).

- This operation assumes mass density (in g/cm^3) and relative electron density (dimensionless; relative to electron density of water, which is $3.343E23$ cm^3) are numerically equivalent. This assumption appears to be reasonable for bulk human tissue (arXiv:1508.00226v1).

3.152.3 Parameters

- ImageSelection
- Filename
- SourcePosition
- AttenuationScale
- ImageModel
- Rows
- Columns

3.152.3.1 ImageSelection

3.152.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.152.3.1.2 Default

- `"last"`

3.152.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.152.3.2 Filename

3.152.3.2.1 Description The filename (or full path) to which the simulated image will be saved to. The format is FITS. Leaving empty will result in a unique name being generated.

3.152.3.2.2 Default

- `""`

3.152.3.2.3 Examples

- `""`
- `"./img.fits"`
- `"sim_radiograph.fits"`
- `"/tmp/out.fits"`

3.152.3.3 SourcePosition

3.152.3.3.1 Description This parameter controls where the virtual point source is. Both absolute and relative positioning are available. A source located at point (1.0, -2.3, 4.5) in the DICOM coordinate system of a given image can be specified as 'absolute(1.0, -2.3, 4.5)'. A source located relative to the image centre by offset (10.0, -23.4, 45.6) in the DICOM coordinate system of a given image can be specified as 'relative(10.0, -23.4, 45.6)'. Relative offsets must be specified relative to the image centre. Note that DICOM units (i.e., mm) are used for all coordinates.

3.152.3.3.2 Default

- `"relative(0.0, 1000.0, 20.0)"`

3.152.3.3.3 Examples

- `"relative(0.0, 1610.0, 20.0)"`
- `"absolute(-123.0, 123.0, 1.23)"`

3.152.3.4 AttenuationScale

3.152.3.4.1 Description This parameter globally scales all attenuation factors derived via ray marching. Adjusting this parameter will alter the radiograph image contrast the exponential attenuation model; numbers within (0:1) will result in less attenuation, whereas numbers within (1:inf) will result in more attenuation. Thin or low-mass subjects might require artifically increased attenuation, whereas thick or high-mass subjects might require artifically decreased attenuation. Setting this number to 1 will result in no scaling. This parameter has units 1/length, and the magnitude should *roughly* correspond with the inverse of about 3× the length transited by a typical ray (in mm).

3.152.3.4.2 Default

- `"0.001"`

3.152.3.4.3 Examples

- `"1.0E-4"`
- `"0.001"`
- `"0.01"`
- `"0.1"`
- `"1.0"`
- `"10.0"`
- `"1E2"`

3.152.3.5 ImageModel

3.152.3.5.1 Description This parameter adjusts how the final image is constructed. As rays transit a voxel, the approximate transit distance is multiplied with the voxel's attenuation coefficient (i.e., $\mu \cdot dL$) to give the ray's attenuation. The sum of all per-voxel attenuations constitutes the total attenuation. There are many ways this information can be converted into an image. First, the 'attenuation-length' model directly outputs the total attenuation for each ray. The simulated image's pixels will contain the total attenuation for one ray. It will almost always provide an image since the attenutation is not performed. This can be thought of as a log transform of a standard radiograph. Second, the 'exponential' model performs the attenuation assuming the radiation beam is monoenergetic, narrow, and has the same energy spectrum as the original imaging device. This model produces a typical radiograph, where each image pixel contains $1 - \exp - \sum \mu \cdot dL$. Note that the values will all $\in [0 : 1]$ (i.e.,

Hounsfield units are *not* used). The overall contrast can be adjusted using the AttenuationScale parameter, however it is easiest to assess a reasonable tuning factor by inspecting the image produced by the 'attenutation-length' model.

3.152.3.5.2 Default

- `"attenuation-length"`

3.152.3.5.3 Supported Options

- `"attenuation-length"`
- `"exponential"`

3.152.3.6 Rows

3.152.3.6.1 Description The number of rows that the simulated radiograph will contain. Note that the field of view is determined separately from the number of rows and columns, so increasing the row count will only result in increased spatial resolution.

3.152.3.6.2 Default

- `"512"`

3.152.3.6.3 Examples

- `"100"`
- `"500"`
- `"2000"`

3.152.3.7 Columns

3.152.3.7.1 Description The number of columns that the simulated radiograph will contain. Note that the field of view is determined separately from the number of rows and columns, so increasing the column count will only result in increased spatial resolution.

3.152.3.7.2 Default

- `"512"`

3.152.3.7.3 Examples

- `"100"`
- `"500"`
- `"2000"`

3.153 SpatialBlur

3.153.1 Description

This operation blurs pixels (within the plane of the image only) using the specified estimator.

3.153.2 Parameters

- ImageSelection
- Estimator
- GaussianOpenSigma

3.153.2.1 ImageSelection

3.153.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.153.2.1.2 Default

- `"all"`

3.153.2.1.3 Examples

- `"last"`
- `"first"`

- "all"
- "none"
- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.153.2.2 Estimator

3.153.2.2.1 Description Controls the (in-plane) blur estimator to use. Options are currently: box_3x3, box_5x5, gaussian_3x3, gaussian_5x5, and gaussian_open. The latter (gaussian_open) is adaptive and requires a supplementary parameter that controls the number of adjacent pixels to consider. The former ('...3x3' and '...5x5') are 'fixed' estimators that use a convolution kernel with a fixed size (3x3 or 5x5 pixel neighbourhoods). All estimators operate in 'pixel-space' and are ignorant about the image spatial extent. All estimators are normalized, and thus won't significantly affect the pixel magnitude scale.

3.153.2.2.2 Default

- "gaussian_open"

3.153.2.2.3 Examples

- "box_3x3"
- "box_5x5"
- "gaussian_3x3"
- "gaussian_5x5"
- "gaussian_open"

3.153.2.3 GaussianOpenSigma

3.153.2.3.1 Description Controls the number of neighbours to consider (only) when using the gaussian_open estimator. The number of pixels is computed automatically to accommodate the specified sigma (currently ignored pixels have 3*sigma or less weighting). Be aware this operation can take an enormous amount of time, since the pixel neighbourhoods quickly grow large.

3.153.2.3.2 Default

- "1.5"

3.153.2.3.3 Examples

- `"0.5"`
- `"1.0"`
- `"1.5"`
- `"2.5"`
- `"5.0"`

3.154 SpatialDerivative

3.154.1 Description

This operation estimates various partial derivatives (of pixel values) within 2D images.

3.154.2 Parameters

- ImageSelection
- Estimator
- Method

3.154.2.1 ImageSelection

3.154.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the

order specified.

3.154.2.1.2 Default

- `"last"`

3.154.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.154.2.2 Estimator

3.154.2.2.1 Description Controls the finite-difference partial derivative order or estimator used. All estimators are centred and use mirror boundary conditions. First-order estimators include the basic nearest-neighbour first derivative, and Roberts' cross, Prewitt, Sobel, Scharr estimators. 'XxY' denotes the size of the convolution kernel (i.e., the number of adjacent pixels considered). The only second-order estimator is the basic nearest-neighbour second derivative.

3.154.2.2.2 Default

- `"Scharr-3x3"`

3.154.2.2.3 Supported Options

- `"first"`
- `"Roberts-cross-3x3"`
- `"Prewitt-3x3"`
- `"Sobel-3x3"`
- `"Sobel-5x5"`
- `"Scharr-3x3"`
- `"Scharr-5x5"`
- `"second"`

3.154.2.3 Method

3.154.2.3.1 Description Controls partial derivative method. First-order derivatives can be row- or column-aligned, Roberts' cross can be (+row,+col)-aligned or (-row,+col)-aligned. Second-order derivatives can be row-aligned, column-aligned, or 'cross' –meaning the compound partial derivative. All methods support non-maximum-suppression for edge thinning, but currently only the magnitude is output. All methods support magnitude (addition of orthogonal components in quadrature) and orientation (in radians; [0,2pi)).

3.154.2.3.2 Default

- `"magnitude"`

3.154.2.3.3 Supported Options

- `"row-aligned"`
- `"column-aligned"`
- `"prow-pcol-aligned"`
- `"nrow-pcol-aligned"`
- `"magnitude"`
- `"orientation"`
- `"non-maximum-suppression"`
- `"cross"`

3.155 SpatialSharpen

3.155.1 Description

This operation 'sharpens' pixels (within the plane of the image only) using the specified estimator.

3.155.2 Parameters

- ImageSelection
- Estimator

3.155.2.1 ImageSelection

3.155.2.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based

indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.155.2.1.2 Default

- `"all"`

3.155.2.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.155.2.2 Estimator

3.155.2.2.1 Description Controls the (in-plane) sharpening estimator to use. Options are currently: sharpen_3x3 and unsharp_mask_5x5. The latter is based on a 5x5 Gaussian blur estimator.

3.155.2.2.2 Default

- `"unsharp_mask_5x5"`

3.155.2.2.3 Supported Options

- `"sharpen_3x3"`
- `"unsharp_mask_5x5"`

3.156 SubdivideSurfaceMeshes

3.156.1 Description

This operation subdivides existing surface meshes according to the specified criteria, replacing the original meshes with subdivided copies.

3.156.2 Notes

- Selected surface meshes should represent polyhedra.

3.156.3 Parameters

- MeshSelection
- Iterations

3.156.3.1 MeshSelection

3.156.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.156.3.1.2 Default

- `"last"`

3.156.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.156.3.2 Iterations

3.156.3.2.1 Description The number of times subdivision should be performed.

3.156.3.2.2 Default

- `"2"`

3.156.3.2.3 Examples

- `"1"`
- `"2"`
- `"5"`

3.157 SubsegmentContours

3.157.1 Description

This operation sub-segments the selected contours, resulting in contours with reduced size.

3.157.2 Parameters

- ROILabelRegex
- NormalizedROILabelRegex
- PlanarOrientation
- ReplaceAllWithSubsegment
- RetainSubsegment
- SubsegMethod
- NestedCleaveOrder
- XSelection
- YSelection
- ZSelection
- FractionalTolerance
- MaxBisects

3.157.2.1 ROILabelRegex

3.157.2.1.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.157.2.1.2 Default

- `".*"`

3.157.2.1.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.157.2.2 NormalizedROILabelRegex

3.157.2.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.157.2.2.2 Default

- `".*"`

3.157.2.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.157.2.3 PlanarOrientation

3.157.2.3.1 Description A string instructing how to orient the cleaving planes. Currently supported: (1) 'axis-aligned' (i.e., align with the image/dose grid row and column unit vectors) and (2) 'static-oblique' (i.e., same as axis-aligned but rotated 22.5 degrees to reduce colinearity, which sometimes improves sub-segment area consistency).

3.157.2.3.2 Default

- `"axis-aligned"`

3.157.2.3.3 Supported Options

- `"axis-aligned"`
- `"static-oblique"`

3.157.2.4 ReplaceAllWithSubsegment

3.157.2.4.1 Description Keep the sub-segment and remove any existing contours from the original ROIs. This is most useful for further processing, such as nested sub-segmentation. Note that sub-segment contours currently have identical metadata to their parent contours.

3.157.2.4.2 Default

- `"false"`

3.157.2.4.3 Examples

- `"true"`
- `"false"`

3.157.2.5 RetainSubsegment

3.157.2.5.1 Description Keep the sub-segment as part of the original ROIs. The contours are appended to the original ROIs, but the contour ROIName and NormalizedROIName are set to the argument provided. (If no argument is provided, sub-segments are not retained.) This is most useful for inspection of sub-segments. Note that sub-segment contours currently have identical metadata to their parent contours, except they are renamed accordingly.

3.157.2.5.2 Default

- `""`

3.157.2.5.3 Examples

- `"subsegment_01"`
- `"subsegment_02"`
- `"selected_subsegment"`

3.157.2.6 SubsegMethod

3.157.2.6.1 Description The method to use for sub-segmentation. Nested sub-segmentation should almost always be preferred unless you know what you're doing. It should be faster too. Compound sub-segmentation is known to cause problems, e.g., with zero-area sub-segments and spatial dependence in sub-segment volume.

3.157.2.6.2 Default

- `"nested-cleave"`

3.157.2.6.3 Supported Options

- `"nested-cleave"`
- `"compound-cleave"`

3.157.2.7 NestedCleaveOrder

3.157.2.7.1 Description The order in which to apply nested cleaves. Typically this will be one of 'ZXX', 'ZYX', 'XYZ', 'XZY', 'YZX', or 'YXZ', but any non-empty combination of 'X', 'Y', and 'Z' are possible. Cleaves are implemented from left to right using the specified X, Y, and Z selection criteria. Multiple cleaves along the same axis are possible, but note that currently the same selection criteria are used for each iteration.

3.157.2.7.2 Default

- `"ZXY"`

3.157.2.7.3 Examples

- `"ZXY"`
- `"ZYX"`
- `"X"`
- `"XYX"`

3.157.2.8 XSelection

3.157.2.8.1 Description (See ZSelection description.) The 'X' direction is defined in terms of movement on an image when the row number increases. This is generally VERTICAL and DOWNWARD for a patient in head-first supine orientation, but it varies with orientation conventions. All selections are defined in terms of the original ROIs.

3.157.2.8.2 Default

- `"1.0;0.0"`

3.157.2.8.3 Examples

- `"0.50;0.50"`
- `"0.50;0.0"`
- `"0.30;0.0"`
- `"0.30;0.70"`

3.157.2.9 YSelection

3.157.2.9.1 Description (See ZSelection description.) The 'Y' direction is defined in terms of movement on an image when the column number increases. This is generally HORIZONTAL and RIGHTWARD for a patient in head-first supine orientation, but it varies with orientation conventions. All selections are defined in terms of the original ROIs.

3.157.2.9.2 Default

- `"1.0;0.0"`

3.157.2.9.3 Examples

- `"0.50;0.50"`
- `"0.50;0.0"`
- `"0.30;0.0"`
- `"0.30;0.70"`

3.157.2.10 ZSelection

3.157.2.10.1 Description The thickness and offset defining the single, continuous extent of the sub-segmentation in terms of the fractional area remaining above a plane. The planes define the portion extracted and are determined such that sub-segmentation will give the desired fractional planar areas. The numbers specify the thickness and offset from the bottom of the ROI volume to the bottom of the extent. The 'upper' direction is take from the contour plane orientation and assumed to be positive if pointing toward the positive-z direction. Only a single 3D selection can be made per operation invocation. Sub-segmentation can be performed in transverse ('Z'), row_unit ('X'), and column_unit ('Y') directions (in that order). All selections are defined in terms of the original ROIs. Note that impossible selections will likely result in errors, e.g., specifying a small constraint when the . Note that it is possible to perform nested sub-segmentation (including passing along the original contours) by opting to replace the original ROI contours with this sub-segmentation and invoking this operation again with the desired sub-segmentation. Examples: If you want the middle 50% of an ROI, specify '0.50;0.25'. If you want the upper 50% then specify '0.50;0.50'. If you want the lower 50% then specify '0.50;0.0'. If you want the upper 30% then specify '0.30;0.70'. If you want the lower 30% then specify '0.30;0.70'.

3.157.2.10.2 Default

- `"1.0;0.0"`

3.157.2.10.3 Examples

- `"0.50;0.50"`
- `"0.50;0.0"`
- `"0.30;0.0"`

- "0.30;0.70"

3.157.2.11 FractionalTolerance

3.157.2.11.1 Description The tolerance of X, Y, and Z fractional area bisection criteria (see ZSelection description). This parameter specifies a stopping condition for the bisection procedure. If it is set too high, sub-segments may be inadequatly rough. If it is set too low, bisection below the machine precision floor may be attempted, which will result in instabilities. Note that the number of permitted iterations will control whether this tolerance can possibly be reached; if strict adherence is required, set the maximum number of iterations to be excessively large.

3.157.2.11.2 Default

- "0.001"

3.157.2.11.3 Examples

- "1E-2"
- "1E-3"
- "1E-4"
- "1E-5"

3.157.2.12 MaxBisects

3.157.2.12.1 Description The maximum number of iterations the bisection procedure can perform. This parameter specifies a stopping condition for the bisection procedure. If it is set too low, sub-segments may be inadequatly rough. If it is set too high, bisection below the machine precision floor may be attempted, which will result in instabilities. Note that the fractional tolerance will control whether this tolerance can possibly be reached; if an exact number of iterations is required, set the fractional tolerance to be excessively small.

3.157.2.12.2 Default

- "20"

3.157.2.12.3 Examples

- "10"
- "20"
- "30"

3.158 Subsegment_ComputeDose_VanLuijk

3.158.1 Description

This operation sub-segments the selected ROI(s) and computes dose within the resulting sub-segments.

3.158.2 Parameters

- AreaDataFileName
- DerivativeDataFileName
- DistributionDataFileName
- NormalizedROILabelRegex
- PlanarOrientation
- ReplaceAllWithSubsegment
- RetainSubsegment
- ROILabelRegex
- SubsegMethod
- XSelection
- YSelection
- ZSelection
- FractionalTolerance
- MaxBisects

3.158.2.1 AreaDataFileName

3.158.2.1.1 Description A filename (or full path) in which to append sub-segment areaa data generated by this routine. The format is CSV. Note that if a sub-segment has zero area or does not exist, no area will be printed. You'll have to manually add sub-segments with zero area as needed if this info is relevant to you (e.g., if you are deriving a population average). Leave empty to NOT dump anything.

3.158.2.1.2 Default

- `""`

3.158.2.1.3 Examples

- `""`
- `"/tmp/somefile"`
- `"localfile.csv"`
- `"area_data.csv"`

3.158.2.2 DerivativeDataFileName

3.158.2.2.1 Description A filename (or full path) in which to append derivative data generated by this routine. The format is CSV. Leave empty to dump to generate a unique temporary file.

3.158.2.2.2 Default

- ""

3.158.2.2.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "derivative_data.csv"

3.158.2.3 DistributionDataFileName

3.158.2.3.1 Description A filename (or full path) in which to append raw distribution data generated by this routine. The format is one line of description followed by one line for the distribution; pixel intensities are listed with a single space between elements; the descriptions contain the patient ID, ROIName, and subsegment description (guaranteed) and possibly various other data afterward. Leave empty to NOT dump anything.

3.158.2.3.2 Default

- ""

3.158.2.3.3 Examples

- ""
- "/tmp/somefile"
- "localfile.csv"
- "distributions.data"

3.158.2.4 NormalizedROILabelRegex

3.158.2.4.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.158.2.4.2 Default

- `".*"`

3.158.2.4.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.158.2.5 PlanarOrientation

3.158.2.5.1 Description A string instructing how to orient the cleaving planes. Currently only 'AxisAligned' (i.e., align with the image/dose grid row and column unit vectors) and 'StaticOblique' (i.e., same as AxisAligned but rotated 22.5 degrees to reduce colinearity, which sometimes improves sub-segment area consistency).

3.158.2.5.2 Default

- `"AxisAligned"`

3.158.2.5.3 Supported Options

- `"AxisAligned"`
- `"StaticOblique"`

3.158.2.6 ReplaceAllWithSubsegment

3.158.2.6.1 Description Keep the sub-segment and remove any existing contours from the original ROIs. This is most useful for further processing, such as nested sub-segmentation. Note that sub-segment contours currently have identical metadata to their parent contours.

3.158.2.6.2 Default

- `"false"`

3.158.2.6.3 Examples

- `"true"`
- `"false"`

3.158.2.7 RetainSubsegment

3.158.2.7.1 Description Keep the sub-segment as part of the original ROIs. The contours are appended to the original ROIs, but the contour ROIName and NormalizedROIName are set to the argument provided. (If no argument is provided, sub-segments are not retained.) This is most useful for inspection of sub-segments. Note that sub-segment contours currently have identical metadata to their parent contours, except they are renamed accordingly.

3.158.2.7.2 Default

- `""`

3.158.2.7.3 Examples

- `"subsegment_01"`
- `"subsegment_02"`
- `"selected_subsegment"`

3.158.2.8 ROILabelRegex

3.158.2.8.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.158.2.8.2 Default

- `".*"`

3.158.2.8.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`

- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.158.2.9 SubsegMethod

3.158.2.9.1 Description The method to use for sub-segmentation. Nested sub-segmentation should almost always be preferred unless you know what you're doing. It should be faster too. The compound method was used in the van Luijk paper, but it is known to have serious problems.

3.158.2.9.2 Default

- "nested"

3.158.2.9.3 Supported Options

- "nested"
- "compound"

3.158.2.10 XSelection

3.158.2.10.1 Description (See ZSelection description.) The "X" direction is defined in terms of movement on an image when the row number increases. This is generally VERTICAL and DOWNWARD. All selections are defined in terms of the original ROIs.

3.158.2.10.2 Default

- "1.0;0.0"

3.158.2.10.3 Examples

- "0.50;0.50"
- "0.50;0.0"
- "0.30;0.0"
- "0.30;0.70"

3.158.2.11 YSelection

3.158.2.11.1 Description (See ZSelection description.) The "Y" direction is defined in terms of movement on an image when the column number increases. This is generally HORIZONTAL and RIGHTWARD. All selections are defined in terms of the original ROIs.

3.158.2.11.2 Default

- `"1.0;0.0"`

3.158.2.11.3 Examples

- `"0.50;0.50"`
- `"0.50;0.0"`
- `"0.30;0.0"`
- `"0.30;0.70"`

3.158.2.12 ZSelection

3.158.2.12.1 Description The thickness and offset defining the single, continuous extent of the sub-segmentation in terms of the fractional area remaining above a plane. The planes define the portion extracted and are determined such that sub-segmentation will give the desired fractional planar areas. The numbers specify the thickness and offset from the bottom of the ROI volume to the bottom of the extent. The 'upper' direction is take from the contour plane orientation and assumed to be positive if pointing toward the positive-z direction. Only a single 3D selection can be made per operation invocation. Sub-segmentation can be performed in transverse ("Z"), row_unit ("X"), and column_unit ("Y") directions (in that order). All selections are defined in terms of the original ROIs. Note that it is possible to perform nested sub-segmentation (including passing along the original contours) by opting to replace the original ROI contours with this sub-segmentation and invoking this operation again with the desired sub-segmentation. If you want the middle 50% of an ROI, specify '0.50;0.25'. If you want the upper 50% then specify '0.50;0.50'. If you want the lower 50% then specify '0.50;0.0'. If you want the upper 30% then specify '0.30;0.70'. If you want the lower 30% then specify '0.30;0.70'.

3.158.2.12.2 Default

- `"1.0;0.0"`

3.158.2.12.3 Examples

- `"0.50;0.50"`
- `"0.50;0.0"`
- `"0.30;0.0"`
- `"0.30;0.70"`

3.158.2.13 FractionalTolerance

3.158.2.13.1 Description The tolerance of X, Y, and Z fractional area bisection criteria (see ZSelection description). This parameter specifies a stopping condition for the bisection procedure. If it is set too high, sub-segments may be inadequatly rough. If it is set too low, bisection below the machine precision floor may be attempted, which will result in instabilities. Note that the number of permitted iterations will control whether this tolerance can possibly be reached; if strict adherence is required, set the maximum number of iterations to be excessively large.

3.158.2.13.2 Default

- `"0.001"`

3.158.2.13.3 Examples

- `"1E-2"`
- `"1E-3"`
- `"1E-4"`
- `"1E-5"`

3.158.2.14 MaxBisects

3.158.2.14.1 Description The maximum number of iterations the bisection procedure can perform. This parameter specifies a stopping condition for the bisection procedure. If it is set too low, sub-segments may be inadequatly rough. If it is set too high, bisection below the machine precision floor may be attempted, which will result in instabilities. Note that the fractional tolerance will control whether this tolerance can possibly be reached; if an exact number of iterations is required, set the fractional tolerance to be excessively small.

3.158.2.14.2 Default

- `"20"`

3.158.2.14.3 Examples

- `"10"`
- `"20"`
- `"30"`

3.159 SubtractImages

3.159.1 Description

This routine subtracts images that spatially overlap.

3.159.2 Notes

- The ReferenceImageSelection is subtracted from the ImageSelection and the result is stored in ImageSelection. So this operation implements $A = A - B$ where A is ImageSelection and B is ReferenceImageSelection. The ReferenceImageSelection images are not altered.
- Multiple image volumes can be selected by both ImageSelection and ReferenceImageSelection. For each ImageSelection volume, each of the ReferenceImageSelection volumes are subtracted sequentially.

3.159.3 Parameters

- ImageSelection
- ReferenceImageSelection

3.159.3.1 ImageSelection

3.159.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.159.3.1.2 Default

- `"last"`

3.159.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.159.3.2 ReferenceImageSelection

3.159.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.159.3.2.2 Default

- `"!last"`

3.159.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.160 SupersampleImageGrid

3.160.1 Description

This operation scales supersamples images so they have more rows and/or columns, but the whole image keeps its shape and spatial extent. This operation is typically used for zooming into images or trying to ensure a sufficient number of voxels are within small contours.

3.160.2 Notes

- Be aware that specifying large multipliers (or even small multipliers on large images) will consume much memory. It is best to pre-crop images to a region of interest if possible.

3.160.3 Parameters

- ImageSelection
- RowScaleFactor
- ColumnScaleFactor
- SliceScaleFactor
- SamplingMethod

3.160.3.1 ImageSelection

3.160.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based

indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.160.3.1.2 Default

- `"last"`

3.160.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.160.3.2 RowScaleFactor

3.160.3.2.1 Description A positive integer specifying how many rows will be in the new images. The number is relative to the incoming image row count. Specifying '1' will result in nothing happening. Specifying '8' will result in 8x as many rows.

3.160.3.2.2 Default

- `"2"`

3.160.3.2.3 Examples

- "1"
- "2"
- "3"
- "8"

3.160.3.3 ColumnScaleFactor

3.160.3.3.1 Description A positive integer specifying how many columns will be in the new images. The number is relative to the incoming image column count. Specifying '1' will result in nothing happening. Specifying '8' will result in 8x as many columns.

3.160.3.3.2 Default

- "2"

3.160.3.3.3 Examples

- "1"
- "2"
- "3"
- "8"

3.160.3.4 SliceScaleFactor

3.160.3.4.1 Description A positive integer specifying how many image slices will be in the new images. The number is relative to the incoming image slice count. Specifying '1' will result in nothing happening. Specifying '8' will result in 8x as many slices. Note that slice supersampling always happens *after* in-plane supersampling. Also note that merely setting this factor will not enable 3D supersampling; you also need to specify a 3D-aware SamplingMethod.

3.160.3.4.2 Default

- "2"

3.160.3.4.3 Examples

- "1"
- "2"
- "3"
- "8"

3.160.3.5 SamplingMethod

3.160.3.5.1 Description The supersampling method to use. Note: 'inplane-' methods only consider neighbours in the plane of a single image – neighbours in adjacent images are not considered and the supersampled image will contain the same number of image slices as the inputs.

3.160.3.5.2 Default

- `"inplane-bilinear"`

3.160.3.5.3 Supported Options

- `"inplane-bicubic"`
- `"inplane-bilinear"`
- `"trilinear"`

3.161 SurfaceBasedRayCastDoseAccumulate

3.161.1 Description

This routine uses rays (actually: line segments) to estimate point-dose on the surface of an ROI. The ROI is approximated by surface mesh and rays are passed through. Dose is interpolated at the intersection points and intersecting lines (i.e., where the ray 'glances' the surface) are discarded. The surface reconstruction can be tweaked, but appear to reasonably approximate the ROI contours; both can be output to compare visually. Though it is not required by the implementation, only the ray-surface intersection nearest to the detector is considered. All other intersections (i.e., on the far side of the surface mesh) are ignored. This routine is fairly fast compared to the slow grid-based counterpart previously implemented. The speedup comes from use of an AABB-tree to accelerate intersection queries and avoid having to 'walk' rays step-by-step through over/through the geometry.

3.161.2 Parameters

- TotalDoseMapFileName
- RefCroppedTotalDoseMapFileName
- IntersectionCountMapFileName
- DepthMapFileName
- RadialDistMapFileName
- RefIntersectionCountMapFileName
- ROISurfaceMeshFileName
- SubdividedROISurfaceMeshFileName
- RefSurfaceMeshFileName
- SubdividedRefSurfaceMeshFileName
- ROICOMCOMLineFileName
- NormalizedReferenceROILabelRegex
- NormalizedROILabelRegex

- ReferenceROILabelRegex
- ROILabelRegex
- SourceDetectorRows
- SourceDetectorColumns
- MeshingSubdivisionIterations
- MaxRaySurfaceIntersections
- OnlyGenerateSurface

3.161.2.1 TotalDoseMapFileName

3.161.2.1.1 Description A filename (or full path) for the total dose image map (at all ray-surface intersection points). The dose for each ray is summed over all ray-surface point intersections. The format is FITS. This file is always generated. Leave the argument empty to generate a unique filename.

3.161.2.1.2 Default

- `""`

3.161.2.1.3 Examples

- `""`
- `"total_dose_map.fits"`
- `"/tmp/out.fits"`

3.161.2.2 RefCroppedTotalDoseMapFileName

3.161.2.2.1 Description A filename (or full path) for the total dose image map (at all ray-surface intersection points). The dose for each ray is summed over all ray-surface point intersections. Doses in this map are only registered when the ray intersects the reference ROI mesh. The format is FITS. This file is always generated. Leave the argument empty to generate a unique filename.

3.161.2.2.2 Default

- `""`

3.161.2.2.3 Examples

- `""`
- `"total_dose_map.fits"`
- `"/tmp/out.fits"`

3.161.2.3 IntersectionCountMapFileName

3.161.2.3.1 Description A filename (or full path) for the (number of ray-surface intersections) image map. Each pixel in this map (and the total dose map) represents a single ray; the number of times the ray intersects the surface can be useful for various purposes, but most often it will simply be a sanity check for the cross-sectional shape or that a specific number of intersections were recorded in regions with geometrical folds. Pixels will all be within [0,MaxRaySurfaceIntersections]. The format is FITS. Leave empty to dump to generate a unique filename.

3.161.2.3.2 Default

- ""

3.161.2.3.3 Examples

- ""
- "intersection_count_map.fits"
- "/tmp/out.fits"

3.161.2.4 DepthMapFileName

3.161.2.4.1 Description A filename (or full path) for the distance (depth) of each ray-surface intersection point from the detector. Has DICOM coordinate system units. This image is potentially multi-channel with MaxRaySurfaceIntersections channels (when MaxRaySurfaceIntersections = 1 there is 1 channel). The format is FITS. Leaving empty will result in no file being written.

3.161.2.4.2 Default

- ""

3.161.2.4.3 Examples

- ""
- "depth_map.fits"
- "/tmp/out.fits"

3.161.2.5 RadialDistMapFileName

3.161.2.5.1 Description A filename (or full path) for the distance of each ray-surface intersection point from the line joining reference and target ROI centre-of-masses. This helps quantify position in 3D. Has DICOM coordinate system units. This image is potentially multi-channel with MaxRaySurfaceIntersections channels (when MaxRaySurfaceIntersections = 1 there is 1 channel). The format is FITS. Leaving empty will result in no file being written.

3.161.2.5.2 Default

- ""

3.161.2.5.3 Examples

- ""
- "radial_dist_map.fits"
- "/tmp/out.fits"

3.161.2.6 RefIntersectionCountMapFileName

3.161.2.6.1 Description A filename (or full path) for the (number of ray-surface intersections) for the reference ROIs. Each pixel in this map (and the total dose map) represents a single ray; the number of times the ray intersects the surface can be useful for various purposes, but most often it will simply be a sanity check for the cross-sectional shape or that a specific number of intersections were recorded in regions with geometrical folds. Note: currently, the number of intersections is limited to 0 or 1! The format is FITS. Leave empty to dump to generate a unique filename.

3.161.2.6.2 Default

- ""

3.161.2.6.3 Examples

- ""
- "ref_roi_intersection_count_map.fits"
- "/tmp/out.fits"

3.161.2.7 ROISurfaceMeshFileName

3.161.2.7.1 Description A filename (or full path) for the (pre-subdivided) surface mesh that is contructed from the ROI contours. The format is OFF. This file is mostly useful for inspection of the surface or comparison with contours. Leaving empty will result in no file being written.

3.161.2.7.2 Default

- ""

3.161.2.7.3 Examples

- ""
- "/tmp/roi_surface_mesh.off"
- "roi_surface_mesh.off"

3.161.2.8 SubdividedROISurfaceMeshFileName

3.161.2.8.1 Description A filename (or full path) for the Loop-subdivided surface mesh that is contructed from the ROI contours. The format is OFF. This file is mostly useful for inspection of the surface or comparison with contours. Leaving empty will result in no file being written.

3.161.2.8.2 Default

- ""

3.161.2.8.3 Examples

- ""
- "/tmp/subdivided_roi_surface_mesh.off"
- "subdivided_roi_surface_mesh.off"

3.161.2.9 RefSurfaceMeshFileName

3.161.2.9.1 Description A filename (or full path) for the (pre-subdivided) surface mesh that is contructed from the reference ROI contours. The format is OFF. This file is mostly useful for inspection of the surface or comparison with contours. Leaving empty will result in no file being written.

3.161.2.9.2 Default

- ""

3.161.2.9.3 Examples

- ""
- "/tmp/roi_surface_mesh.off"
- "roi_surface_mesh.off"

3.161.2.10 SubdividedRefSurfaceMeshFileName

3.161.2.10.1 Description A filename (or full path) for the Loop-subdivided surface mesh that is contructed from the reference ROI contours. The format is OFF. This file is mostly useful for inspection of the surface or comparison with contours. Leaving empty will result in no file being written.

3.161.2.10.2 Default

- ""

3.161.2.10.3 Examples

- ""
- "/tmp/subdivided_roi_surface_mesh.off"
- "subdivided_roi_surface_mesh.off"

3.161.2.11 ROICOMCOMLineFileName

3.161.2.11.1 Description A filename (or full path) for the line segment that connected the centre-of-mass (COM) of reference and target ROI. The format is OFF. This file is mostly useful for inspection of the surface or comparison with contours. Leaving empty will result in no file being written.

3.161.2.11.2 Default

- ""

3.161.2.11.3 Examples

- ""
- "/tmp/roi_com_com_line.off"
- "roi_com_com_line.off"

3.161.2.12 NormalizedReferenceROILabelRegex

3.161.2.12.1 Description A regex matching reference ROI labels/names to consider. The default will match all available ROIs, which is non-sensical. The reference ROI is used to orient the cleaving plane to trim the grid surface mask.

3.161.2.12.2 Default

- ".*"

3.161.2.12.3 Examples

- ".*"
- ".*Prostate.*"
- "Left Kidney"
- "Gross Liver"

3.161.2.13 NormalizedROILabelRegex

3.161.2.13.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI

name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.161.2.13.2 Default

- `".*"`

3.161.2.13.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.161.2.14 ReferenceROILabelRegex

3.161.2.14.1 Description A regex matching reference ROI labels/names to consider. The default will match all available ROIs, which is non-sensical. The reference ROI is used to orient the cleaving plane to trim the grid surface mask.

3.161.2.14.2 Default

- `".*"`

3.161.2.14.3 Examples

- `".*"`
- `".*[pP]rostate.*"`
- `"body"`
- `"Gross_Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.161.2.15 ROILabelRegex

3.161.2.15.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.161.2.15.2 Default

- `".*"`

3.161.2.15.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.161.2.16 SourceDetectorRows

3.161.2.16.1 Description The number of rows in the resulting images, which also defines how many rays are used. (Each pixel in the source image represents a single ray.) Setting too fine relative to the surface mask grid or dose grid is futile.

3.161.2.16.2 Default

- `"1024"`

3.161.2.16.3 Examples

- `"100"`
- `"128"`
- `"1024"`
- `"4096"`

3.161.2.17 SourceDetectorColumns

3.161.2.17.1 Description The number of columns in the resulting images. (Each pixel in the source image represents a single ray.) Setting too fine relative to the surface mask grid or dose grid is futile.

3.161.2.17.2 Default

- "1024"

3.161.2.17.3 Examples

- "100"
- "128"
- "1024"
- "4096"

3.161.2.18 MeshingSubdivisionIterations

3.161.2.18.1 Description The number of iterations of Loop's subdivision to apply to the surface mesh. The aim of subdivision in this context is to have a smooth surface to work with, but too many applications will create too many facets. More facets will not lead to more precise results beyond a certain (modest) amount of smoothing. If the geometry is relatively spherical already, and meshing bounds produce reasonably smooth (but 'blocky') surface meshes, then 2-3 iterations should suffice. More than 3-4 iterations will almost always be inappropriate.

3.161.2.18.2 Default

- "2"

3.161.2.18.3 Examples

- "0"
- "1"
- "2"
- "3"

3.161.2.19 MaxRaySurfaceIntersections

3.161.2.19.1 Description The maximum number of ray-surface intersections to accumulate before retiring each ray. Note that intersections are sorted spatially by their distance to the detector, and those closest to the detector are considered first. If the ROI surface is opaque, setting this value to 1 will emulate visibility. Setting to 2 will permit rays continue through the ROI and pass through the other side; dose will be the accumulation of dose at each ray-surface intersection. This value should most often be 1 or some very high number (e.g.,

1000) to make the surface either completely opaque or completely transparent. (A transparent surface may help to visualize geometrical 'folds' or other surface details of interest.)

3.161.2.19.2 Default

- `"1"`

3.161.2.19.3 Examples

- `"1"`
- `"4"`
- `"1000"`

3.161.2.20 OnlyGenerateSurface

3.161.2.20.1 Description Stop processing after writing the surface and subdivided surface meshes. This option is primarily used for debugging and visualization.

3.161.2.20.2 Default

- `"false"`

3.161.2.20.3 Examples

- `"true"`
- `"false"`

3.162 ThresholdImages

3.162.1 Description

This operation applies thresholds to images. Both upper and lower thresholds can be specified.

3.162.2 Notes

- This routine operates on individual images. When thresholds are specified on a percentile basis, each image is considered separately and therefore each image may be thresholded with different values.
- Both thresholds are inclusive. To binarize an image, use the same threshold for both upper and lower threshold parameters. Voxels that fall on the threshold will currently be treated as if they exclusively satisfy the upper threshold, but this behaviour is not guaranteed.

3.162.3 Parameters

- Lower
- Low
- Upper
- High
- Channel
- ImageSelection

3.162.3.1 Lower

3.162.3.1.1 Description The lower bound (inclusive). Pixels with values < this number are replaced with the 'low' value. If this number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If this number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile).

3.162.3.1.2 Default

- `"-inf"`

3.162.3.1.3 Examples

- `"0.0"`
- `"-1E-99"`
- `"1.23"`
- `"0.2%"`
- `"23tile"`
- `"23.123 tile"`

3.162.3.2 Low

3.162.3.2.1 Description The value a pixel will take when below the lower threshold.

3.162.3.2.2 Default

- `"-inf"`

3.162.3.2.3 Examples

- `"0.0"`
- `"-1000.0"`
- `"-inf"`
- `"nan"`

3.162.3.3 Upper

3.162.3.3.1 Description The upper bound (inclusive). Pixels with values > this number are replaced with the 'high' value. If this number is followed by a '%', the bound will be scaled between the min and max pixel values [0-100%]. If this number is followed by 'tile', the bound will be replaced with the corresponding percentile [0-100tile]. Note that upper and lower bounds can be specified separately (e.g., lower bound is a percentage, but upper bound is a percentile).

3.162.3.3.2 Default

- `"inf"`

3.162.3.3.3 Examples

- `"1.0"`
- `"1E-99"`
- `"2.34"`
- `"98.12%"`
- `"94tile"`
- `"94.123 tile"`

3.162.3.4 High

3.162.3.4.1 Description The value a pixel will take when above the upper threshold.

3.162.3.4.2 Default

- `"inf"`

3.162.3.4.3 Examples

- `"0.0"`
- `"1000.0"`
- `"inf"`
- `"nan"`

3.162.3.5 Channel

3.162.3.5.1 Description The image channel to use. Zero-based.

3.162.3.5.2 Default

- `"0"`

3.162.3.5.3 Examples

- `"0"`
- `"1"`
- `"2"`

3.162.3.6 ImageSelection

3.162.3.6.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.162.3.6.2 Default

- `"last"`

3.162.3.6.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`

- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.163 ThresholdOtsu

3.163.1 Description

This routine performs Otsu thresholding (i.e., 'binarization') on an image volume. The thresholding is limited within ROI(s). Otsu thresholding works best on images with a well-defined bimodal voxel intensity histogram. It works by finding the threshold that partitions the voxel intensity histogram into two parts, essentially so that the sum of each partition's variance is minimal. The number of histogram bins (i.e., number of distinct voxel magnitude levels) is configurable. Voxels are binarized; the replacement values are also configurable.

3.163.2 Notes

- The Otsu method will not necessarily cleanly separate bimodal peaks in the voxel intensity histogram.

3.163.3 Parameters

- ImageSelection
- HistogramBins
- ReplacementLow
- ReplacementHigh
- OverwriteVoxels
- Channel
- NormalizedROILabelRegex
- ROILabelRegex
- ContourOverlap
- Inclusivity

3.163.3.1 ImageSelection

3.163.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.163.3.1.2 Default

- `"last"`

3.163.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.163.3.2 HistogramBins

3.163.3.2.1 Description The number of equal-width bins the histogram should have. Classically, images were 8-bit integer-valued and thus 255 bins were commonly used. However, because floating-point numbers are used practically any number of bins are supported. What is optimal (or acceptable) depends on the analytical requirements. If the threshold does not have to be exact, try use the smallest number of bins you can get away with; 50-150 should suffice. This will speed up computation. If the threshold is being used for analytical purposes, use as many bins as the data can support – if the voxel values span only 8-bit integers, having more than 255 bins will not improve the analysis. Likewise if voxels are discretized or sparse. Experiment by gradually increasing the number of bins until the threshold value converges to a reasonable number, and then use that number of bins for future analysis.

3.163.3.2.2 Default

- "255"

3.163.3.2.3 Examples

- "10"
- "50"
- "100"
- "200"
- "500"

3.163.3.3 ReplacementLow

3.163.3.3.1 Description The value to give voxels which are below (exclusive) the Otsu threshold value.

3.163.3.3.2 Default

- "0.0"

3.163.3.3.3 Examples

- "-1.0"
- "0.0"
- "1.23"
- "nan"
- "inf"

3.163.3.4 ReplacementHigh

3.163.3.4.1 Description The value to give voxels which are above (inclusive) the Otsu threshold value.

3.163.3.4.2 Default

- "1.0"

3.163.3.4.3 Examples

- "-1.0"
- "0.0"
- "1.23"
- "nan"
- "inf"

3.163.3.5 OverwriteVoxels

3.163.3.5.1 Description Controls whether voxels should actually be binarized or not. Whether or not voxel intensities are overwritten, the Otsu threshold value is written into the image metadata as 'OtsuThreshold' in case further processing is needed.

3.163.3.5.2 Default

- `"true"`

3.163.3.5.3 Examples

- `"true"`
- `"false"`

3.163.3.6 Channel

3.163.3.6.1 Description The image channel to use. Zero-based.

3.163.3.6.2 Default

- `"0"`

3.163.3.6.3 Examples

- `"0"`
- `"1"`
- `"2"`

3.163.3.7 NormalizedROILabelRegex

3.163.3.7.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.163.3.7.2 Default

- `".*"`

3.163.3.7.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.163.3.8 ROILabelRegex

3.163.3.8.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.163.3.8.2 Default

- ".*"

3.163.3.8.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.163.3.9 ContourOverlap

3.163.3.9.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.163.3.9.2 Default

- `"ignore"`

3.163.3.9.3 Supported Options

- `"ignore"`
- `"honour_opposite_orientations"`
- `"overlapping_contours_cancel"`
- `"honour_opps"`
- `"overlap_cancel"`

3.163.3.10 Inclusivity

3.163.3.10.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.163.3.10.2 Default

- `"center"`

3.163.3.10.3 Supported Options

- `"center"`
- `"centre"`
- `"planar_corner_inclusive"`
- `"planar_inc"`
- `"planar_corner_exclusive"`
- `"planar_exc"`

3.164 TrimROIDose

3.164.1 Description

This operation provides a simplified interface for overriding the dose within a ROI. For example, this operation can be used to modify a base plan by eliminating dose that coincides with a PTV/CTV/GTV/ROI etc.

3.164.2 Notes

- This operation performs the opposite of the 'Crop' operation, which trims the dose outside a ROI.
- The inclusivity of a dose voxel that straddles the ROI boundary can be specified in various ways. Refer to the Inclusivity parameter documentation.
- By default this operation only overrides dose within a ROI. The opposite, overriding dose outside of a ROI, can be accomplished using the expert interface.

3.164.3 Parameters

- Channel
- ImageSelection
- ContourOverlap
- Inclusivity
- Method
- ExteriorVal
- InteriorVal
- ExteriorOverwrite
- InteriorOverwrite
- NormalizedROILabelRegex
- ROILabelRegex
- ImageSelection
- Filename
- ParanoiaLevel

3.164.3.1 Channel

3.164.3.1.1 Description The image channel to use. Zero-based. Use '-1' to operate on all available channels.

3.164.3.1.2 Default

- `"-1"`

3.164.3.1.3 Examples

- `"-1"`
- `"0"`
- `"1"`
- `"2"`

3.164.3.2 ImageSelection

3.164.3.2.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.164.3.2.2 Default

- `"all"`

3.164.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.164.3.3 ContourOverlap

3.164.3.3.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of

contour orientation. This will effectively honour only the outermost contour regardless of orientation, but provides the most predictable and consistent results. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. This is useful for Boolean structures where contour orientation is significant for interior contours (holes). If contours do not have consistent overlap (e.g., if contours intersect) the results can be unpredictable and hard to interpret. The option 'overlapping_contours_cancel' ignores orientation and alternately cancerls all overlapping contours. Again, if the contours do not have consistent overlap (e.g., if contours intersect) the results can be unpredictable and hard to interpret.

3.164.3.3.2 Default

- `"ignore"`

3.164.3.3.3 Supported Options

- `"ignore"`
- `"honour_opposite_orientations"`
- `"overlapping_contours_cancel"`
- `"honour_opps"`
- `"overlap_cancel"`

3.164.3.4 Inclusivity

3.164.3.4.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.164.3.4.2 Default

- `"planar_inc"`

3.164.3.4.3 Supported Options

- `"center"`
- `"centre"`
- `"planar_corner_inclusive"`
- `"planar_inc"`
- `"planar_corner_exclusive"`
- `"planar_exc"`

3.164.3.5 Method

3.164.3.5.1 Description Controls the type of image mask that is generated. The default, 'binary', exclusively overwrites voxels with the InteriorValue or ExteriorValue. Another method is 'receding_squares' which creates a mask which, if processed with the marching-squares algorithm, will (mostly) recreate the original contours. The 'receding_squares' can be considered the inverse of the marching-squares algorithm. Note that the 'receding_squares' implementation is not optimized for speed.

3.164.3.5.2 Default

- `"binary"`

3.164.3.5.3 Supported Options

- `"binary"`
- `"receding_squares"`

3.164.3.6 ExteriorVal

3.164.3.6.1 Description The value to give to voxels outside the specified ROI(s). For the 'binary' method, note that this value will be ignored if exterior overwrites are disabled. For the 'receding_squares' method this value is used to define the threshold needed to recover the original contours (mean of InteriorVal and ExteriorVal).

3.164.3.6.2 Default

- `"0.0"`

3.164.3.6.3 Examples

- `"0.0"`
- `"-1.0"`
- `"1.23"`
- `"2.34E26"`

3.164.3.7 InteriorVal

3.164.3.7.1 Description The value to give to voxels within the specified ROI(s). For the 'binary' method, note that this value will be ignored if interior overwrites are disabled. For the 'receding_squares' method this value is used to define the threshold needed to recover the original contours (mean of InteriorVal and ExteriorVal).

3.164.3.7.2 Default

- `"0.0"`

3.164.3.7.3 Examples

- `"0.0"`
- `"-1.0"`
- `"1.23"`
- `"2.34E26"`

3.164.3.8 ExteriorOverwrite

3.164.3.8.1 Description Whether to overwrite voxels exterior to the specified ROI(s).

3.164.3.8.2 Default

- `"false"`

3.164.3.8.3 Examples

- `"true"`
- `"false"`

3.164.3.9 InteriorOverwrite

3.164.3.9.1 Description Whether to overwrite voxels interior to the specified ROI(s).

3.164.3.9.2 Default

- `"true"`

3.164.3.9.3 Examples

- `"true"`
- `"false"`

3.164.3.10 NormalizedROILabelRegex

3.164.3.10.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.164.3.10.2 Default

- `".*"`

3.164.3.10.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.164.3.11 ROILabelRegex

3.164.3.11.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.164.3.11.2 Default

- `".*"`

3.164.3.11.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.164.3.12 ImageSelection

3.164.3.12.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.164.3.12.2 Default

- `"all"`

3.164.3.12.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.164.3.13 Filename

3.164.3.13.1 Description The filename (or full path name) to which the DICOM file should be written.

3.164.3.13.2 Default

- `"/tmp/RD.dcm"`

3.164.3.13.3 Examples

- `"/tmp/RD.dcm"`
- `"./RD.dcm"`
- `"RD.dcm"`

3.164.3.14 ParanoiaLevel

3.164.3.14.1 Description At low paranoia setting, only top-level UIDs are replaced. At medium paranoia setting, many UIDs, descriptions, and labels are replaced, but the PatientID and FrameOfReferenceUID are retained. The high paranoia setting is the same as the medium setting, but the PatientID and FrameOfReferenceUID are also replaced. (Note: this is not a full anonymization.) Use the low setting if you want to retain linkage to the originating data set. Use the medium setting if you don't. Use the high setting if your TPS goes overboard linking data sets by PatientID and/or FrameOfReferenceUID.

3.164.3.14.2 Default

- `"medium"`

3.164.3.14.3 Supported Options

- `"low"`
- `"medium"`
- `"high"`

3.165 UBC3TMRI_DCE

3.165.1 Description

This operation is used to generate dynamic contrast-enhanced MRI contrast enhancement maps.

3.165.2 Parameters

No registered options.

3.166 UBC3TMRI_DCE_Differences

3.166.1 Description

This operation is used to generate dynamic contrast-enhanced MRI contrast enhancement maps.

3.166.2 Notes

- This routine generates difference maps using both long DCE scans. Thus it takes up a LOT of memory! Try avoid unnecessary copies of large (temporally long) arrays.

3.166.3 Parameters

No registered options.

3.167 UBC3TMRI_DCE_Experimental

3.167.1 Description

This operation is an experimental operation for processing dynamic contrast-enhanced MR images.

3.167.2 Parameters

No registered options.

3.168 UBC3TMRI_IVIM_ADC

3.168.1 Description

This operation is an experimental operation for processing IVIM MR images into ADC maps.

3.168.2 Parameters

No registered options.

3.169 VolumetricCorrelationDetector

3.169.1 Description

This operation can assess 3D correlations by sampling the neighbourhood surrounding each voxel and assigning a similarity score. This routine is useful for

detecting repetitive (regular) patterns that are known in advance.

3.169.2 Notes

- The provided image collection must be rectilinear.
- At the moment this routine can only be modified via recompilation.

3.169.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Low
- High
- Channel

3.169.3.1 ImageSelection

3.169.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.169.3.1.2 Default

- `"last"`

3.169.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.169.3.2 NormalizedROILabelRegex

3.169.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.169.3.2.2 Default

- `".*"`

3.169.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.169.3.3 ROILabelRegex

3.169.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.169.3.3.2 Default

- `".*"`

3.169.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.169.3.4 Low

3.169.3.4.1 Description The low percentile.

3.169.3.4.2 Default

- `"0.05"`

3.169.3.4.3 Examples

- `"0.05"`
- `"0.5"`
- `"0.99"`

3.169.3.5 High

3.169.3.5.1 Description The high percentile.

3.169.3.5.2 Default

- `"0.95"`

3.169.3.5.3 Examples

- `"0.95"`
- `"0.5"`
- `"0.05"`

3.169.3.6 Channel

3.169.3.6.1 Description The channel to operated on (zero-based). Negative values will cause all channels to be operated on.

3.169.3.6.2 Default

- `"-1"`

3.169.3.6.3 Examples

- `"-1"`
- `"0"`
- `"1"`

3.170 VolumetricSpatialBlur

3.170.1 Description

This operation performs blurring of voxel values within 3D rectilinear image arrays.

3.170.2 Notes

- The provided image collection must be rectilinear.

3.170.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Channel
- Estimator

3.170.3.1 ImageSelection

3.170.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.170.3.1.2 Default

- `"last"`

3.170.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.170.3.2 NormalizedROILabelRegex

3.170.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.170.3.2.2 Default

- `".*"`

3.170.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.170.3.3 ROILabelRegex

3.170.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.170.3.3.2 Default

- `".*"`

3.170.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`

- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.170.3.4 Channel

3.170.3.4.1 Description The channel to operated on (zero-based). Negative values will cause all channels to be operated on.

3.170.3.4.2 Default

- "-1"

3.170.3.4.3 Examples

- "-1"
- "0"
- "1"

3.170.3.5 Estimator

3.170.3.5.1 Description Controls which type of blur is computed. Currently, 'Gaussian' refers to a fixed sigma=1 (in pixel coordinates, not DICOM units) Gaussian blur that extends for 3*sigma thus providing a 7x7x7 window. Note that applying this kernel N times will approximate a Gaussian with sigma=N. Also note that boundary voxels will cause accessible voxels within the same window to be more heavily weighted. Try avoid boundaries or add extra margins if possible.

3.170.3.5.2 Default

- "Gaussian"

3.170.3.5.3 Supported Options

- "Gaussian"

3.171 VolumetricSpatialDerivative

3.171.1 Description

This operation estimates various spatial partial derivatives (of pixel values) within 3D rectilinear image arrays.

3.171.2 Notes

- The provided image collection must be rectilinear.

3.171.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- Channel
- Estimator
- Method

3.171.3.1 ImageSelection

3.171.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.171.3.1.2 Default

- `"last"`

3.171.3.1.3 Examples

- `"last"`
- `"first"`

- "all"
- "none"
- "#0"
- "#-0"
- "!last"
- "!#-3"
- "key@.*value.*"
- "key1@.*value1.*;key2@^value2$;first"
- "numerous"

3.171.3.2 NormalizedROILabelRegex

3.171.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.171.3.2.2 Default

- ".*"

3.171.3.2.3 Examples

- ".*"
- ".*Body.*"
- "Body"
- "liver"
- ".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"
- "Left Parotid|Right Parotid"

3.171.3.3 ROILabelRegex

3.171.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI

name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.171.3.3.2 Default

- `".*"`

3.171.3.3.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.171.3.4 Channel

3.171.3.4.1 Description The channel to operated on (zero-based). Negative values will cause all channels to be operated on.

3.171.3.4.2 Default

- `"-1"`

3.171.3.4.3 Examples

- `"-1"`
- `"0"`
- `"1"`

3.171.3.5 Estimator

3.171.3.5.1 Description Controls the finite-difference partial derivative order or estimator used. All estimators are centred and use mirror boundary conditions. First-order estimators include the basic nearest-neighbour first derivative and Sobel estimators. 'XxYxZ' denotes the size of the convolution kernel (i.e., the number of adjacent pixels considered).

3.171.3.5.2 Default

- `"Sobel-3x3x3"`

3.171.3.5.3 Supported Options

- `"first"`
- `"Sobel-3x3x3"`

3.171.3.6 Method

3.171.3.6.1 Description Controls partial derivative method. First-order derivatives can be row-, column-, or image-aligned, All methods also support magnitude (addition of orthogonal components in quadrature).

3.171.3.6.2 Default

- `"magnitude"`

3.171.3.6.3 Supported Options

- `"row-aligned"`
- `"column-aligned"`
- `"image-aligned"`
- `"magnitude"`
- `"non-maximum-suppression"`

3.172 VoxelRANSAC

3.172.1 Description

This routine performs RANSAC fitting using voxel positions as inputs. The search can be confined within ROIs and a range of voxel intensities.

3.172.2 Notes

- This operation does not make use of voxel intensities during the RANSAC procedure. Voxel intensities are only used to identify which voxel positions are considered.

3.172.3 Parameters

- ImageSelection
- NormalizedROILabelRegex
- ROILabelRegex
- ContourOverlap
- Inclusivity
- Channel
- Lower
- Upper

- GridSeparation

3.172.3.1 ImageSelection

3.172.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.172.3.1.2 Default

- `"last"`

3.172.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.172.3.2 NormalizedROILabelRegex

3.172.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.172.3.2.2 Default

- `".*"`

3.172.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.172.3.3 ROILabelRegex

3.172.3.3.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.172.3.3.2 Default

- `".*"`

3.172.3.3.3 Examples

- ".*"
- ".*body.*"
- "body"
- "^body$"
- "Liver"
- ".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"
- "left_parotid|right_parotid"

3.172.3.4 ContourOverlap

3.172.3.4.1 Description Controls overlapping contours are treated. The default 'ignore' treats overlapping contours as a single contour, regardless of contour orientation. The option 'honour_opposite_orientations' makes overlapping contours with opposite orientation cancel. Otherwise, orientation is ignored. The latter is useful for Boolean structures where contour orientation is significant for interior contours (holes). The option 'overlapping_contours_cancel' ignores orientation and cancels all contour overlap.

3.172.3.4.2 Default

- "ignore"

3.172.3.4.3 Supported Options

- "ignore"
- "honour_opposite_orientations"
- "overlapping_contours_cancel"
- "honour_opps"
- "overlap_cancel"

3.172.3.5 Inclusivity

3.172.3.5.1 Description Controls how voxels are deemed to be 'within' the interior of the selected ROI(s). The default 'center' considers only the central-most point of each voxel. There are two corner options that correspond to a 2D projection of the voxel onto the image plane. The first, 'planar_corner_inclusive', considers a voxel interior if ANY corner is interior. The second, 'planar_corner_exclusive', considers a voxel interior if ALL (four) corners are interior.

3.172.3.5.2 Default

- "center"

3.172.3.5.3 Supported Options

- "center"
- "centre"
- "planar_corner_inclusive"
- "planar_inc"
- "planar_corner_exclusive"
- "planar_exc"

3.172.3.6 Channel

3.172.3.6.1 Description The channel to operated on (zero-based). Negative values will cause all channels to be operated on.

3.172.3.6.2 Default

- "0"

3.172.3.6.3 Examples

- "-1"
- "0"
- "1"

3.172.3.7 Lower

3.172.3.7.1 Description Lower threshold (inclusive) below which voxels will be ignored by this routine.

3.172.3.7.2 Default

- "-inf"

3.172.3.7.3 Examples

- "-inf"
- "0.0"
- "1024"

3.172.3.8 Upper

3.172.3.8.1 Description Upper threshold (inclusive) above which voxels will be ignored by this routine.

3.172.3.8.2 Default

- "inf"

3.172.3.8.3 Examples

- `"inf"`
- `"1.0"`
- `"2048"`

3.172.3.9 GridSeparation

3.172.3.9.1 Description The known separation of the grid (in DICOM units; mm) being sought.

3.172.3.9.2 Default

- `"nan"`

3.172.3.9.3 Examples

- `"1.0"`
- `"1.5"`
- `"10.0"`
- `"1.23E4"`

3.173 WarpContours

3.173.1 Description

This operation applies a transform object to the specified contours, warping them spatially.

3.173.2 Notes

- A transform object must be selected; this operation cannot create transforms. Transforms can be generated via registration or by parsing user-provided functions.
- Contours are transformed in-place. Metadata may become invalid by this operation.
- This operation can only handle individual transforms. If multiple, sequential transforms are required, this operation must be invoked multiple time. This will guarantee the ordering of the transforms.
- Transformations are not (generally) restricted to the coordinate frame of reference that they were derived from. This permits a single transformation to be applicable to point clouds, surface meshes, images, and contours.

3.173.3 Parameters

- ROILabelRegex
- NormalizedROILabelRegex
- TransformSelection

3.173.3.1 ROILabelRegex

3.173.3.1.1 Description A regular expression (regex) matching *raw* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match 'raw' contour labels.

3.173.3.1.2 Default

- `".*"`

3.173.3.1.3 Examples

- `".*"`
- `".*body.*"`
- `"body"`
- `"^body$"`
- `"Liver"`
- `".*left.*parotid.*|.*right.*parotid.*|.*eyes.*"`
- `"left_parotid|right_parotid"`

3.173.3.2 NormalizedROILabelRegex

3.173.3.2.1 Description A regular expression (regex) matching *normalized* ROI contour labels/names to consider.

Selection is performed on a whole-ROI basis; individual contours cannot be selected. Be aware that input spaces are trimmed to a single space. If your ROI name has more than two sequential spaces, use regular expressions or escaping to avoid them. All ROIs you want to select must match the provided (single) regex, so use boolean or ('|') if needed. The regular expression engine is extended POSIX and is case insensitive. '.*' will match all available ROIs.

Note that this parameter will match contour labels that have been *normalized* (i.e., mapped, translated) using the user-provided provided lexicon. This is useful

for handling data with heterogeneous naming conventions where fuzzy matching is required. Refer to the lexicon for available labels.

3.173.3.2.2 Default

- `".*"`

3.173.3.2.3 Examples

- `".*"`
- `".*Body.*"`
- `"Body"`
- `"liver"`
- `".*Left.*Parotid.*|.*Right.*Parotid.*|.*Eye.*"`
- `"Left Parotid|Right Parotid"`

3.173.3.3 TransformSelection

3.173.3.3.1 Description Select one or more transform objects (aka 'warp' objects). Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth transformation (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last transformation. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the transformation composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all transformation that do not have the greatest number of sub-objects, not the least-numerous transformation (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.173.3.3.2 Default

- `"last"`

3.173.3.3.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`

3.174 WarpImages

3.174.1 Description

This operation applies a transform object to the specified image arrays, warping them spatially.

3.174.2 Notes

- A transform object must be selected; this operation cannot create transforms. Transforms can be generated via registration or by parsing user-provided functions.
- Images are transformed in-place. Metadata may become invalid by this operation.
- This operation can only handle individual transforms. If multiple, sequential transforms are required, this operation must be invoked multiple time. This will guarantee the ordering of the transforms.
- This operation currently supports only affine transformations. Local transformations require special handling and voxel resampling that is not yet implemented.
- Transformations are not (generally) restricted to the coordinate frame of reference that they were derived from. This permits a single transformation to be applicable to point clouds, surface meshes, images, and contours.

3.174.3 Parameters

- ImageSelection
- TransformSelection

3.174.3.1 ImageSelection

3.174.3.1.1 Description Select one or more image arrays. Note that image arrays can hold anything, but will typically represent a single contiguous 3D volume (i.e., a volumetric CT scan) or '4D' time-series. Be aware that it is possible to mix logically unrelated images together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth image array (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last image array. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the image array composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all image array that do not have the greatest number of sub-objects, not the least-numerous image array (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.174.3.1.2 Default

- `"last"`

3.174.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.174.3.2 TransformSelection

3.174.3.2.1 Description Select one or more transform objects (aka 'warp' objects). Selection specifiers can be of three types: positional, metadata-based

key@value regex, and intrinsic.

Positional specifiers can be ‘first’, ‘last’, ‘none’, or ‘all’ literals. Additionally ‘#N’ for some positive integer N selects the Nth transformation (with zero-based indexing). Likewise, ‘#-N’ selects the Nth-from-last transformation. Positional specifiers can be inverted by prefixing with a ‘!’.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a ‘!’). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the ‘numerous’ and ‘fewest’ literals, which selects the transformation composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a ‘!’. Note that ‘!numerous’ means all transformation that do not have the greatest number of sub-objects, not the least-numerous transformation (i.e., ‘fewest’).

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ‘;’ and are applied in the order specified.

3.174.3.2.2 Default

- `"last"`

3.174.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`

3.175 WarpMeshes

3.175.1 Description

This operation applies a transform object to the specified surface meshes, warping them spatially.

3.175.2 Notes

- A transform object must be selected; this operation cannot create transforms. Transforms can be generated via registration or by parsing user-provided functions.
- Meshes are transformed in-place. Metadata may become invalid by this operation.
- This operation can only handle individual transforms. If multiple, sequential transforms are required, this operation must be invoked multiple time. This will guarantee the ordering of the transforms.
- Transformations are not (generally) restricted to the coordinate frame of reference that they were derived from. This permits a single transformation to be applicable to point clouds, surface meshes, images, and contours.

3.175.3 Parameters

- MeshSelection
- TransformSelection

3.175.3.1 MeshSelection

3.175.3.1.1 Description Select one or more surface meshes. Note that a single surface mesh may hold many disconnected mesh components; they should collectively represent a single logically cohesive object. Be aware that it is possible to mix logically unrelated sub-meshes together in a single mesh. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth surface mesh (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last surface mesh. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the surface mesh composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all surface mesh that do not have the greatest number of sub-objects, not the least-numerous surface mesh (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.175.3.1.2 Default

- `"last"`

3.175.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.175.3.2 TransformSelection

3.175.3.2.1 Description Select one or more transform objects (aka 'warp' objects). Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth transformation (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last transformation. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the transformation composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all transformation that do not have the greatest number of sub-objects, not the least-numerous transformation (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.175.3.2.2 Default

- `"last"`

3.175.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`

3.176 WarpPoints

3.176.1 Description

This operation applies a transform object to the specified point clouds, warping them spatially.

3.176.2 Notes

- A transform object must be selected; this operation cannot create transforms. Transforms can be generated via registration or by parsing user-provided functions.
- Point clouds are transformed in-place. Metadata may become invalid by this operation.
- This operation can only handle individual transforms. If multiple, sequential transforms are required, this operation must be invoked multiple time. This will guarantee the ordering of the transforms.
- Transformations are not (generally) restricted to the coordinate frame of reference that they were derived from. This permits a single transformation to be applicable to point clouds, surface meshes, images, and contours.

3.176.3 Parameters

- PointSelection
- TransformSelection

3.176.3.1 PointSelection

3.176.3.1.1 Description The point cloud that will be transformed. Select one or more point clouds. Note that point clouds can hold a variety of data with varying attributes, but each point cloud is meant to represent a single

logically cohesive collection of points. Be aware that it is possible to mix logically unrelated points together. Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth point cloud (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last point cloud. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the point cloud composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all point cloud that do not have the greatest number of sub-objects, not the least-numerous point cloud (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.176.3.1.2 Default

- `"last"`

3.176.3.1.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`
- `"numerous"`

3.176.3.2 TransformSelection

3.176.3.2.1 Description The transformation that will be applied. Select one or more transform objects (aka 'warp' objects). Selection specifiers can be of three types: positional, metadata-based key@value regex, and intrinsic.

Positional specifiers can be 'first', 'last', 'none', or 'all' literals. Additionally '#N' for some positive integer N selects the Nth transformation (with zero-based indexing). Likewise, '#-N' selects the Nth-from-last transformation. Positional specifiers can be inverted by prefixing with a '!'.

Metadata-based key@value expressions are applied by matching the keys verbatim and the values with regex. In order to invert metadata-based selectors, the regex logic must be inverted (i.e., you can *not* prefix metadata-based selectors with a '!'). Note regexes are case insensitive and should use extended POSIX syntax.

Intrinsic specifiers are currently limited to the 'numerous' and 'fewest' literals, which selects the transformation composed of the greatest and fewest number of sub-objects. Intrinsic specifiers can be inverted by prefixing with a '!'. Note that '!numerous' means all transformation that do not have the greatest number of sub-objects, not the least-numerous transformation (i.e., 'fewest').

All criteria (positional, metadata, and intrinsic) can be mixed together. Multiple criteria can be specified by separating them with a ';' and are applied in the order specified.

3.176.3.2.2 Default

- `"last"`

3.176.3.2.3 Examples

- `"last"`
- `"first"`
- `"all"`
- `"none"`
- `"#0"`
- `"#-0"`
- `"!last"`
- `"!#-3"`
- `"key@.*value.*"`
- `"key1@.*value1.*;key2@^value2$;first"`

4 Known Issues and Limitations

4.1 Hanging on Debian

The SFML_Viewer operation hangs on some systems after viewing a plot with Gnuplot. This stems from a known issue in Ygor.

4.2 Build Requirements

DICOMautomaton depends on several heavily templated libraries and external projects. It requires a considerable amount of memory to build.

4.3 DICOM-RT Support Incomplete

Support for the DICOM Radiotherapy extensions are limited. In particular, only RTDOSE files can currently be exported, and RTPLAN files are not supported at all. Read support for DICOM image modalities and RTSTRUCTS are generally supported well. Broader DICOM support is planned for a future release.