Growth RTV

Python API Reference

V2.1.06J

2021/06/14

e-Growth Co., Ltd.

目　次

1. 概要 1
2. はじめに 2
3. API リファレンス 3
3.1. readRTDir 3
3.2. getImageDcmTags 3
3.3. getImageParam 3
3.4. readGRTVRaw 4
3.5. getROIList 5
3.6. setNewROI 5
3.7. predNewROI 6
3.8. exportROIMesh 7
3.9. interpROI 7
3.10. setDose 8
3.11. exportRT 8
3.12. replaceCurrentImage 9
3.13. exportAsNewDcm 9
3.14. imageRawToDicom 10
3.15. resizeImage3D 10
3.16. augmentImg3D 11
3.17. deformCurrentImageByMesh 12
3.18. closeGRTVTFSession 12
4. サンプルコード 14
4.1. サンプル 1 – 様々な API 呼び出し 14
4.2. サンプル 2 (ROI 定義および RT 化の一括処理) 16
4.3. サンプル 3(深層学習用 3D Raw の一括抽出) 18
4.4. サンプル 4(深層学習用 3D ボリュームデータの拡張) 20
4.5. サンプル 5(DICOM IMAGE のピクセル置換および出力) 22
4.6. サンプル 6(3D Raw の DICOM 変換および出力) 23

1. 概要

Growth RTV では Python 連携プラットフォームを搭載し、ユーザは Growth RTV にバンドルされている連携用モジュール(grtv)をインポートするだけで、pyGRTV で提供されている様々な API を利用してデータ読み込みから、3D データの抽出・加工を簡便に行えます。本仕様書は、pyGRTV の API 仕様についてまとめたものです。

2. はじめに

まず、Growth RTV のデータへアクセスを行うためには、まず grtv モジュールをインポートし、pyGRTV クラスのインスタンス生成を行います。通常、grtv モジュールは「C:¥Program Files¥GrowthRTV¥pyGRTV¥」以下にインストールされています。インスタンス生成と同時に、pyGRTV は Growth RTV から接続情報を受取、初期化を行います。

例:

```
import sys
import numpy as np

sys.path.append("C:/Program Files/GrowthRTV/pyGRTV")
from grtv import *

# GRTV インスタンス生成
grtv = pyGRTV()
```

インスタンス生成時に接続エラーメッセージが表示されなければ、インスタンス生成処理は成功です。Growth RTVプロセスが起動されていない、またはすでに他のプロセスによってpyGRTVのインスタンスが生成されている場合、エラーメッセージが表示され、Pythonプロセスは終了します。

3. API リファレンス

3.1. readRTDir

readRTDir(rtDirPath)

DICOM(CT / MRI /RT)データの読み込みを行います。

引数

- rtDirPath : DICOM ファイル群が直接保存されているフォルダのパス

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗

3.2. getImageDcmTags

getImageDcmTags(dcmTagsMap)

DICOM イメージデータのタグ(イメージボリュームを構成する 1 枚目のスライス)を読み込みます。

引数

- dcmTagsMap ： map 型のコンテナ。取得した DICOM タグはこのコンテナ内に読み込まれます。

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗

3.3. getImageParam

getImageParam(paramID, paramMap)

現在の画像のサイズ情報を取得します。

引数:

- paramID : 取得したいパラメータの ID。
  - GRTV_PARAM_IMAGE_SIZE : X, Y, Z の各 3 次元方向のボクセル数
  - GRTV_PARAM_IMAGE_RES : X, Y, Z の各 3 次元方向のボクセル実寸(mm)
- paramMap : map 型のコンテナ。取得したパラメータ値が格納されます。

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗
-

3.4. readGRTVRaw

readGRTVRaw(grtvDataType, dataID=-1)

画像データの 3 次元ボリュームデータを 3 次元 numpy 型として取得します。

引数

- grtvDataType: 取得したい画像データタイプの ID。
  - GRTV_DATA_TYPE_IMAGE : CT または MRI のピクセル(輝度値)データ
  - GRTV_DATA_TYPE_DOSE ： 線量分布のピクセル(輝度値)データ。Growth RTV では線量データを CT データまたは MRI データと同じ 3 次元サイズに線形補間したデータを返します。
  - GRTV_DATA_TYPE_ROI : Structure(関心領域)の 3D 領域データ。Growth RTV では RT Structure のコンター情報から CT または MRI データと同じ 3 次元サイズのボリュームに変換して返します。3D 領域データは 0(領域外)と 255(領域内の値を持つ。
- dataID: grtvDataType に GRTV_DATA_TYPE_ROI を指定した場合のみこの引数を利用します。dataID は Structure のインデックスを指定します。

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗
- rawData : 取得された 3 次元 numpy 型のボリュームデータ

3.5. getROIList

getROIList(roiList)

定義済み Structure をリスト型として取得します。

引数

- roiList: リスト型のコンテナ。定義済み Structure のリストが map 型として roiList 内に格納されます。各 Structure の map 内に下記の情報が格納されます。
  - roi_name : Structure 名
  - roi_index : Structure のインデックス。readGRTVRaw API などに利用します。

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗

3.6. setNewROI

setNewROI(roiRaw, roiName, roiColor)

新たな Structure を現在の画像内に追加定義します。追加定義される Structure は領域情報(roiRaw)に従い、体積、3D メッシュおよび 2D コンターが自動計算され、Dose(線量分布)がある場合は DVH も算出・表示されます。

引数

- roiRaw: 0~255 を持つ領域情報。CT または MRI の 3 次元データと同じサイズの 3 次元 Numpy 配列であることが必要です。
- roiName: Structure 名

- roiColor: Structure の表示色。0~255 の整数型で。要素長が 3(R, G, B)のリスト型または Numpy 配列であることが必要です。

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗

3.7. predNewROI

predNewROI(roiName, roiColor)

Growth RTV の臓器自動抽出機能を利用して Structure を追加定義します。追加定義される Structure は体積、3D メッシュおよび 2D コンターが自動計算され、Dose(線量分布)がある場合は DVH も算出・表示されます。なお、この API を利用する場合、自動抽出時は GPU 版の TensorFlow が呼び出されるため、GPU メモリの競合が発生しないよう、ユーザ側の Python スクリプトでは GPU 版の TensorFlow を起動しないことを推奨します。

引数

- roiName: Growth RTV がサポートする臓器名を指定します。現在は以下になります。
  - body : 体表
  - llung : 左肺野
  - rlung : 右肺野
  - spinal_cord : 脊柱管
  - liver : 肝臓
  - lkidney : 左腎臓
  - rkidney : 右腎臓
  - stomach : 胃
- roiColor: Structure の表示色。0~255 の整数型で。要素長が 3(R, G, B)のリスト型または Numpy 配列であることが必要です。

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗

3.8. exportROIMesh

exportROIMesh(dataID, meshPath, meshType=GRTV_MESH_TYPE_STL)

Structure を STL または PLY 形式でファイル出力します。

引数

- dataID : Structure のインデックス
- meshPath : メッシュの出力ファイルパス
- meshType : 出力形式。
  - GRTV_MESH_TYPE_STL : STL 形式
  - GRTV_MESH_TYPE_PLY : PLY 形式

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗

3.9. interpROI

interpROI(self, roiRaw)

スライス間隔を空けて定義された領域情報(roiRAW)を線形補間した結果を取得します。

引数

- roiRaw: 0~255 を持つ領域情報。CT または MRI の 3 次元データと同じサイズの 3 次元 Numpy 配列であることが必要です。

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗
- rawData : 取得された 3 次元 numpy 型のボリュームデータ

3.10. setDose

setDose(doseRaw)

現在の画像内の Dose(線量分布)を変更します。変更された線量分布(doseRaw)に従い、DVH も再計算されます。

引数

- doseRaw: 0~65,535 を持つ cGY(センチグレイ)の領域情報。CT または MRI の 3 次元データと同じサイズの 3 次元 Numpy 配列であることが必要です。

戻り値

- status
  - GRTV_STATUS_OK : 設定成功
  - GRTV_STATUS_FALIED : 設定失敗

3.11. exportRT

exportRT(exportDir)

現在のデータを新たな DICOM-RT 形式としてファイル出力します。

引数

- exportDir: 出力先のディレクトリパス
- meshPath : メッシュの出力ファイルパス
- meshType : 出力形式。
  - GRTV_MESH_TYPE_STL : STL 形式
  - GRTV_MESH_TYPE_PLY : PLY 形式

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗

3.12. replaceCurrentImage

replaceCurrentImage(img3d_in, exportDcmDir="")

現在の読み込まれている DICOM IMAGE のピクセルデータを置き換え、新たなシリーズの DICOM 形式としてのファイル出力を行います。

引数

- img3d_in: 置き換える Numpy 型の 3 次元配列。現在読み込まれている CT または MRI の 3 次元データと同じサイズの 3 次元 Numpy 配列であることが必要です。
- exportDcmDir: 出力先のディレクトリパス。指定しない場合はピクセル値の置き換え処理のみが行われます。指定されている場合、現在読み込まれている DICOM IMAGE に対し､新たなシリーズとして出力されます(新たな Series Instance UID が付与されます)。

戻り値

- status
  - GRTV_STATUS_OK : 処理成功
  - GRTV_STATUS_FALIED : 処理失敗

3.13. exportAsNewDcm

exportAsNewDcm (exportDcmDir)

現在の読み込まれている DICOM IMAGE を新たなシリーズの DICOM 形式としてのファイル出力を行います。

引数

- exportDcmDir: 出力先のディレクトリパス｡現在読み込まれている DICOM IMAGE に対し、新たなシリーズとして出力されます(新たな Series Instance UID が付与されます)。

戻り値

- status
  - GRTV_STATUS_OK : 処理成功
  - GRTV_STATUS_FALIED : 処理失敗

3.14. imageRawToDicom

imageRawToDicom(img3d_in, res_z, res_y, res_x, exportDcmDir):

3D ボリュームデータを DICOM 形式へ変換出力を行います。

引数

- img3d_in: 変換元の Numpy 型の 3 次元配列。int8, uint8, int16 の場合は 2 バイト長の CT データ, uint16 の場合は 2 バイト長の MRI データとして DICOM 形式に変換されます。
- res_z: 3D ボリュームデータの Z 方向のスライス間隔
- res_y: 3D ボリュームデータの Y 方向のピクセル間隔
- res_x: 3D ボリュームデータの X 方向のピクセル間隔
- exportDcmDir: 出力先のディレクトリパス。現在読み込まれている DICOM IMAGE に対し、新たなシリーズとして出力されます(新たな Series Instance UID が付与されます)。

戻り値

- status
  - GRTV_STATUS_OK : 処理成功
  - GRTV_STATUS_FALIED : 処理失敗

3.15. resizeImage3D

resizeImage3D(img3d_in, newShape, dataType=np.int16)
3D ボリュームデータを線形補間します。

引数

- img3d_in: 線形補間元の Numpy 型の 3 次元配列
- newShape: 線形補間先の 3 次元サイズ(タプル型)
- dataType: ボリュームデータの Numpy のデータ型。現在は uint8, int16, uint16 がサポートされています。

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗
- rawData : 線形補間後の 3 次元 numpy 型のボリュームデータ

3.16. augmentImg3D

augmentImg3D(imageList, outputDirectory, augmentTimes=5, fileNameList=[],
rotate=True, rotateAngRange=2.5, deform=False, deformRate=0.25)

3D ボリュームデータに対し、回転および局所変形によるランダムデータ拡張を行います。

引数

- imageList: 拡張元の Numpy 型の 3 次元配列のリスト。本リストに格納された 3 次元配列は整合性を維持しながらデータ拡張されます。
  リスト格納例(CT および臓器領域の同時拡張): [img_CT, img_ROI1, img_ROI2]
- outputDirectory: 拡張データの出力フォルダ:
- augmentTimes: 拡張回数。
- fileNameList: imageList に格納された各 3 次元配列に対し、出力ファイル名を設定可能です。fileNameList を設定済みの場合、指定ファイル名をプレフィックスとし使用して拡張データを出力します。
  例: fileName.0001.raw, fileName1.0002.raw, ...
  fileNameList 未設定の場合、img_000N (N は imageList 内の Index)が設定されます。
  例: img_0000.0001.raw, img_0000.0002.raw, ...
- rotate: データ拡張時の回転有無を指定します。本オプションは deform と併用可能です。
- rotateAngRange: 最大回転角を指定します。拡張元 3D ボリュームデータは指定回転角範囲内において、X、Y、Z 軸中心にランダム回転してデータ拡張されます。
- deform: データ拡張時の変形有無を指定します。本オプションは rotate と併用可能です。
- deformRate: 最大変形量を指定します。3D ボリュームデータは、-1.0～1.0 の範囲の三次元空間内において、上記最大変形量の範囲内でランダム変形してデータ拡張されます。

戻り値

- status
  - GRTV_STATUS_OK : 処理成功
  - GRTV_STATUS_FALIED : 処理失敗

3.17. deformCurrentImageByMesh

deformCurrentImageByMesh(srcPlyList, dstPlyList, deform_resolution)

変形前後の表面メッシュ群を指定して、現在読み込みしている画像データを 3 次元変形します。

引数

- srcPlyList: 変形前の表面メッシュファイル(PLY 形式)のリスト。
  リスト格納例: ["D:/src_body.ply", "D:/src_lung.ply", "D:/src_liver.ply"]
- dstPlyList: 変形後の表面メッシュファイル(PLY 形式)のリスト。srcPlyList とは PLY の順序が必要であり、かつ変形前後の各 PLY 同士の頂点対応が必要です。
- resolution: 変形計算精度を指定します。精度が高いほど、計算時間を要します。また、計算時間は PLY の頂点数および読み込み中の画像データのボクセル数に依存します。
  - GRTV_DEFORM_RESOLUTION_HIGH : 最高精度
  - GRTV_DEFORM_RESOLUTION_MIDDLE： 高精度
  - GRTV_DEFORM_RESOLUTION_STANDARD： 標準精度
  - GRTV_DEFORM_RESOLUTION_LOW：低精度

戻り値

- status
  - GRTV_STATUS_OK : 処理成功
  - GRTV_STATUS_FALIED : 処理失敗

3.18. closeGRTVTFSession

closeGRTVTFSession()

Growth RTV 側の臓器の自動抽出用の TensorFlow セッションを終了します。ユーザ側で TensorFlow を実行したい場合、この API によって先に Growth RTV 側のセッションを終了し、GPU メモリを開放することを推奨します。

引数

- なし

戻り値

- status
  - GRTV_STATUS_OK : 読み込み成功
  - GRTV_STATUS_FALIED : 読み込み失敗

4. サンプルコード

4.1. サンプル 1 – 様々な API 呼び出し

このサンプルでは様々な API の呼び出し例を示します。

```
import numpy as np
from time import sleep
import zmq
import os, sys, glob, shutil
import psutil
import numpy as np
import re, gc
import cv2
sys.path.append("C:/Program Files/GrowthRTV/pyGRTV")
from grtv import *

if __name__ == "__main__":

    # GRTV インスタンス生成
    grtv = pyGRTV()
    print(grtv.pyIni)
    print(grtv.userPyCache)

    # DICOM-RT 読み込み
    rtDir = "c:/testdata/"
    ret = grtv.readRTDir(rtDir)

    #  画像 DICOM タグ取得サンプル
    dcmTagsMap = {}
    ret = grtv.getImageDcmTags(dcmTagsMap)
    print(dcmTagsMap)

    #  現在の画像のサイズ情報取得
    paramMap = {}
```

```
ret = grtv.getImageParam(grtv.GRTV_PARAM_IMAGE_SIZE, paramMap)
print(ret)

#現在の画像のボクセルサイズ取得
ret = grtv.getImageParam(grtv.GRTV_PARAM_IMAGE_RES, paramMap)
print(paramMap)

# CT RAW  ボリューム取得サンプルコード
ret, img_short = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_IMAGE)

# Dose  ボリューム取得サンプルコード
ret, dose_short = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_DOSE)

# ROI リスト取得
roiList = []
ret = grtv.getROIList(roiList)
print(roiList)

#  中間スライス表示サンプルコード
imgShape = img_short.shape
slicePos = int(imgShape[0] / 2)
slice = img_short[slicePos, :, :]
gray = grtv.getSliceByLut(slice, wl=0, ww=2000)
cv2.imshow("slice", gray)
cv2.waitKey(0)

### roi Mesh Export サンプル
if False:
    meshType = grtv.GRTV_MESH_TYPE_PLY
    extMap = {grtv.GRTV_MESH_TYPE_STL:".stl", grtv.GRTV_MESH_TYPE_PLY:".ply"}
    meshDir = "d:/dltest/tmp/mesh"
    if not os.path.isdir(meshDir):
        os.makedirs(meshDir)
    for roi in roiList:
        meshPath = "%s/%s%s" % (meshDir, roi["roi_name"], extMap[meshType])
        ret = grtv.exportROIMesh(dataID=int(roi["roi_index"]), meshPath=meshPath,
```

```
meshType=meshType)
            print(ret)
```

4.2. サンプル 2 (ROI 定義および RT 化の一括処理)

このサンプルでは、複数の CT データに対し、臓器抽出から DICOM-RT 化してエクスポートを一括処理する例を示します。

```
import os, sys, glob, shutil
sys.path.append("C:/Program Files/GrowthRTV/pyGRTV")
from grtv import *

if __name__ == "__main__":

    # GRTV インスタンス生成
    grtv = pyGRTV()

    #  複数症例に対する臓器抽出～RT エクスポートまでのサンプルコード
    dataRoot = "c:/data/CTSets/"
    expDataRoot = "c:/data/export/"
    if not os.path.isdir(expDataRoot):
        os.makedirs(expDataRoot)

    dirList = os.listdir(dataRoot)
    exportedDirList = os.listdir(expDataRoot)
    for dirName in dirList:
        if dirName in exportedDirList or "export" == dirName:
            continue
        print("dirName", dirName)

        #  データ読み込み
        dirPath = dataRoot + dirName
        ret = grtv.readRTDir(dirPath)
        if ret == grtv.GRTV_STATUS_FALIED:
            continue
```

```
# ROI 自動抽出サンプルコード
roiList, definedRoiNames = [], []
ret = grtv.getROIList(roiList)
# for roi in roiList:
#     definedRoiNames.append(roi["roi_name"])
if ("rlung" not in definedRoiNames):
    ret = grtv.predNewROI("rlung", (0, 255, 0))  # right lung 自動抽出
    print(ret)
if ("llung" not in definedRoiNames):
    ret = grtv.predNewROI("llung", (0, 0, 255))  # left lung 自動抽出
    print(ret)

#### merge as rlung and llung as total lung
roiList = []
ret = grtv.getROIList(roiList)
print(roiList)
roiId1, roiId2 = -1, -1
for roi in roiList:
    if (roi["roi_name"] == "rlung"):
        roiId1 = int(roi["roi_index"])
    if (roi["roi_name"] == "llung"):
        roiId2 = int(roi["roi_index"])

if roiId1 < 0 or roiId2 < 0:
    continue
print("read rlung")
ret, roi_uint8_LLung = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_ROI, dataID=roiId1)  # 自動抽出された right lung の領域を取得
print("read llung")
ret, roi_uint8_Heart = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_ROI, dataID=roiId2)  # 自動抽出された left lung の領域を取得
# merge
roi_new = roi_uint8_LLung.astype("int") + roi_uint8_Heart.astype("int")  # 領域をマージ
roi_new = np.clip(roi_new, 0, 255).astype("uint8")
# Set to GRTV
```

```
        print("set merged roi")
        ret = grtv.setNewROI(roi_new, "Lung", (255, 255, 0))   #  マージされた領域を Lung として定義

        #  新たな RT データとしてエクスポート
        print(ret, "export rt")
        exportDir = expDataRoot + dirName
        if not os.path.isdir(exportDir):
            os.makedirs(exportDir)
        ret = grtv.exportRT(exportDir)
```

4.3. サンプル 3(深層学習用 3D Raw の一括抽出)

このサンプルでは、複数症例に対する深層学習用 CT および ROI(教師データ)の 3D Raw の抽出～3D 補間によるダウンサイジング処理～ファイル出力を一括処理する例を示します。

```
import os, sys, glob, shutil
sys.path.append("C:/Program Files/GrowthRTV/pyGRTV")
from grtv import *
import csv

if __name__ == "__main__":

    grtv = pyGRTV()        # GRTV インスタンス生成

    #  複数症例に対する深層学習用 CT および ROI(教師データ)の 3D Raw 出力サンプル
    #  本サンプルでは 3D Raw の出力の際に、3 次元補完処理によるダウンサイジング処理の例を含む
    dataRoot = "c:/data/RTSets/"
    expDataRoot = "c:/data/raw3d/"
    if not os.path.isdir(expDataRoot):
        os.makedirs(expDataRoot)

    #  データフォルダー覧取得
    dirList = os.listdir(dataRoot)
    dirList.sort()
```

```
    # 各データについて出力処理
    for iData, dirName in enumerate(dirList):
        print("dirName", dirName)

        dirPath = dataRoot + os.sep + dirName
        ret = grtv.readRTDir(dirPath)          # データ読み込み
        if ret == grtv.GRTV_STATUS_FALIED:
            continue

        print( "read ct" )
        ret, img_short = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_IMAGE)      # CT データ読み込み
        newShape = (img_short.shape[0], 256, 256)

        fInfoCsv = "%s/%04d_info.txt" % (expDataRoot, iData)
        with open(fInfoCsv, 'w') as f:
            writer = csv.writer(f, lineterminator='¥n')
            writer.writerow(list(newShape))        # サイズ情報のファイル保存
            if False:
                # ボクセル実寸情報をファイルに保存する場合
                paramMap = {}
                ret = grtv.getImageParam(grtv.GRTV_PARAM_IMAGE_RES, paramMap)
                res_z = float(paramMap["res_z"])
                res_y = float(paramMap["res_y"]) * img_short.shape[1] / 256
                res_x = float(paramMap["res_x"]) * img_short.shape[2] / 256
                writer.writerow([res_z, res_y, res_x])

        print("reisize ct")
        # CT の 3D データを(originalZ, 256,256)にリサイズ
        ret, img_short_resize = grtv.resizeImage3D(img_short, newShape, dataType=np.int16)
        resizeCTPath = "%s/%04d_ct.raw" % (expDataRoot, iData)
        print( "write reisized ct")
        grtv.writeRaw(resizeCTPath, img_short_resize, dataType=np.int16)      # リサイズした CT 3D
Raw をファイル保存

        # 深層学習用に各 ROI の 3D RAW を書き出し
```

```
        # (3D 線形補間によるリサイズ処理のサンプル含む)
        roiList = []
        ret = grtv.getROIList(roiList)
        print(roiList)
        for roi in roiList:
            print(roi["roi_index"], roi["roi_name"])
            # ROI の 3D Raw を取得
            ret, roi_uint8 = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_ROI,
dataID=int(roi["roi_index"]))

            # (originalZ, 256,256)にリサイズ
            ret, roi_uint8_resize = grtv.resizeImage3D(roi_uint8, newShape, dataType=np.uint8)

            #  リサイズしたROI 3D Raw をファイル保存
            resizeRoiPath = "%s/%04d_%s.raw" % (expDataRoot, iData, roi["roi_name"])
            grtv.writeRaw(resizeRoiPath, roi_uint8_resize, dataType=np.uint8)
```

4.4. サンプル 4(深層学習用 3D ボリュームデータの拡張)

このサンプルでは、深層学習用 CT および ROI(教師データ)の 3D Raw データ拡張処理する例を示します。拡張されたデータ群は raw 形式で指定フォルダ内に出力されます。本サンプルで読み込んでいる DICOM-RT データでは関心領域(ROI)として 0 番目に Body、5 番目に LLung が定義されており、CT とこれらの ROI の整合性を維持した上でデータ拡張する例を示します。

```
import sys
sys.path.append("C:/Program Files/GrowthRTV/pyGRTV")
from grtv import *

if __name__ == "__main__":

    # GRTV インスタンス生成
    grtv = pyGRTV()
```

```
# データ読み込み
rtDir = "d:/testdata"
ret = grtv.readRTDir(rtDir)

# CT RAW ボリューム取得
print("read CT")
ret, ct_short = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_IMAGE)

# 0 番目の ROI(Body)の領域を取得
print("read Body")
ret, roi_uint8_Body = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_ROI, dataID=0)

# 5 番目の ROI(LLung)の領域を取得
print("read LLung")
ret, roi_uint8_LLung = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_ROI, dataID=5)

# 拡張データリスト(本リストのデータ群は整合性を維持してデータ拡張されます)
imageList = [ct_short, roi_uint8_Body, roi_uint8_LLung]

# 各データ名リスト
fileNameList = ["ct", "body", "llung"]

print("augmentImg3D")
ret = grtv.augmentImg3D(imageList=imageList,
                        outputDirectory="d:/augImgs",
                        augmentTimes=5,       #5 回拡張を実施
                        fileNameList=fileNameList,
                        rotate=True,      #回転有り
                        rotateAngRange=3.0,       #最大回転角は 3 度
                        deform=True,      #変形有り
                        deformRate=0.05   #最大変形量は 0.05
                        )
```

4.5. サンプル 5(DICOM IMAGE のピクセル置換および出力)

このサンプルでは、読み込まれた CT データに対し、ピクセルの置き換え処理および出力処理する例を示します。

```
import os, sys, glob, shutil
sys.path.append("C:/Program Files/GrowthRTV/pyGRTV")
from grtv import *

if __name__ == "__main__":

    # GRTV インスタンス生成
    grtv = pyGRTV()

    # DICOM 読み込み
    rtDir = "d:/convTest/dcm0"
    ret = grtv.readRTDir(rtDir)
    print("read dicom", ret)

    # CT RAW ボリューム取得
    ret, img_short = grtv.readGRTVRaw(grtv.GRTV_DATA_TYPE_IMAGE)
    print("get 3d raw", ret)

    orgShape = img_short.shape
    img_short[int(orgShape[0]*3/4):] = 500      #上部 1/4 の CT 値を 500 に設定

    print("call replace image API")

    # replace と export の同時実行
    ret = grtv.replaceCurrentImage(img_short, exportDcmDir="d:/convTest/repDcm1/")
    print("replace result1", ret)

    img_short[:int(orgShape[0] / 4)] = 200   # 更に下部 1/4 の CT 値を 200 に設定

    # replace と export を分けて実行
```

```
    ret = grtv.replaceCurrentImage(img_short)
    print("replace result2", ret)
    grtv.exportAsNewDcm(exportDcmDir="d:/convTest/repDcm2/")
    print("export result", ret)
```

4.6. サンプル 6(3D Raw の DICOM 変換および出力)

このサンプルでは、指定した 3D ボリュームデータ(raw)を DICOM Image への変換処理する例を示します。

```
import os, sys, glob, shutil
sys.path.append("C:/Program Files/GrowthRTV/pyGRTV")
from grtv import *

if __name__ == "__main__":

    # GRTV インスタンス生成
    grtv = pyGRTV()

    # 3D Raw  読み込み
    img_short = grtv.readRaw("d:/convTest/test.raw", dataType=np.int16)

    # 3 次元配列(depth, height, width) に整形
    img_short = img_short.reshape((512, 512, 512))

    #  画素間隔
    resZ, resY, resX = 1.5, 1.0, 1.0

    # Dicom へ変換
    grtv.imageRawToDicom(img_short, resZ, resY, resX, "d:/convTest/rawToDcm")
```