Exporting Contours to DICOM-RT Structure Set

Release 1.00

Subrahmanyam Gorthi, Meritxell Bach Cuadra and Jean-Philippe Thiran

January 29, 2009

Signal Processing Laboratory (LTS5), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. {subrahmanyam.gorthi, meritxell.bach, jp.thiran}@epfl.ch

Abstract

This paper presents an ITK [1] implementation for exporting the contours of the automated segmentation results to DICOM-RT Structure Set format. The "radiotherapy structure set" (RTSTRUCT) object of the DICOM standard is used for the transfer of patient structures and related data, between the devices found within and outside the radiotherapy department. It mainly contains the information of regions of interest (ROIs) and points of interest (E.g. dose reference points). In many cases, rather than manually drawing these ROIs on the CT images, one can indeed benefit from the automated segmentation algorithms already implemented in ITK. But at present, it is not possible to export the ROIs obtained from ITK to RTSTRUCT format. In order to bride this gap, we have developed a framework for exporting contour data to RTSTRUCT. We provide here the complete implementation of RTSTRUCT exporter and present the details of the pipeline used. Results on a 3-D CT image of the Head and Neck (H&N) region are presented.

Contents

1   Introduction

"Digital Imaging and Communications in Medicine" (DICOM) is a standard for handling, storing, printing and transmitting information in medical imaging. DICOM enables the integration of scanners, servers, workstations, printers and network hardware from multiple manufacturers into a picture archiving and communication system (PACS). It supports a wide variety of modules like CT, MR, nuclear medicine, ultrasound, X-ray, PET and *radiotherapy (RT)*.

There are several RT-specific modules in DICOM standard; the key DICOM-RT modules are RT Image, RT Dose, *RTSTRUCT*, RT Plan and RT Treatment Record [1]. DICOM-RT, through these modules, is found to effectively describe most of the data needed in radiotherapy. Hence, major manufacturers have already adopted this standard for their applications. This paper deals with exclusively the RTSTRUCT module, in the context of writing contour data.

The purpose of the RTSTRUCT module is to address the requirements for transfer of patient structures and related data defined on CT scanners, virtual simulation workstations, treatment planning systems and similar devices. RTSTRUCT module is described in more detail in the next Section.

The rest of the paper is organized as follows: The next Section describes various modules contained by the RTSTRUCT and their attributes. The implementation details are presented in Section 3. Results on a 3-D CT image of the H&N region are presented in Section 4. Finally, conclusions and future work are presented in Section 5.

2   RTSTRUCT

[2] contains a comprehensive documentation of the DICOM standard; this documentation is vast (spread over 14 documents with more than 2000 pages!) and covers all the modules of the DICOM in detail. Our present work, however, is limited to only the RTSTRUCT and its related modules; further, we use only those attributes of the modules that are necessary for creating a valid and meaningful RTSTRUCT file. Hence, for an easy reference, we present here a summary of the RTSTRUCT [2].

The mandatory modules contained by the RTSTRUCT are presented in Table 1. These modules are grouped based on the type of information entity that they represent. We now present a brief description of each of

Table 1: Mandatory modules of RTSTRUCT.

Information Entity (IE)	Mandatory Modules
Patient	(i) Patient
Study	(ii) General Study
Series	(iii) RT Series
Equipment	(iv) General Equipment
Structure Set	(v) Structure Set
	(vi) ROI Contour
	(vii) RT ROI Observations
	(viii) SOP Common

these modules and mention how their attributes are obtained for creating the RTSTRUCT file:

---

[1]Unlike many other DICOM modules (E.g. CT, MR, PET), DICOM-RT has mostly non-image information.

[2] The summary presented is based on the authors' understanding of the DICOM documentation [2] and thus, it is prone to errors.

(i) Patient Module: This module specifies the attributes of the patient that describe and identify the patient who is the subject of a diagnostic study. This module contains attributes of the patient that are needed for diagnostic interpretation of the image and are common for all studies performed on the patient. Table 2 presents the attributes that we use for creating the RTSTRUCT file. A brief description of each attribute along with its DICOM tag are presented in the Table. Patient module is common to both CT image and RT-

Table 2: Patient module attributes that we use for writing the RTSRTUCT file.

S.No.	Attribute Name	Tag	Attribute Description
1	Patient's Name	(0010, 0010)	Patient's full name.
2	Patient ID	(0010, 0020)	Hospital ID number for the patient.
3	Patient's Birth Date	(0010, 0030)	Birth date of the patient.
4	Patient's Sex	(0010, 0040)	Sex of the named patient.

STRUCT; hence, we extract this information from an input DICOM CT image for creating the RTSTRUCT file.

(ii) General Study Module: This module specifies the attributes that describe and identify the study performed upon the patient. Table 3 presents the attributes that we use for creating the RTSTRUCT file.

A small note on the notation followed: Certain Tables describe sequences of items by using the symbol: '>'. The symbol '>' precedes the attribute name of the members of an *item*. All marked attributes belong to the generic description of an item which may be repeated to form a sequence of items. This sequence of items is nested in the attribute which precedes in the table the first member marked with a '>'. For an easy tracing of sequences, a sub-numbering is used in the S.No. column of the Tables.

Table 3: General study module attributes that we use for writing the RTSRTUCT file.

S.No	Attribute Name	Tag	Attribute Description
1	Study Instance UID	(0020, 000d)	Unique identifier for the Study.
2	Study Date	(0008, 0020)	Date the Study started.
3	Study Time	(0008, 0030)	Time the Study started.
4	Referring Physician's Name	(0008, 0090)	Name of the patient's ref. physician.
5	Study ID	(0020, 0010)	Study identifier.
6	Accession Number	(0008, 0050)	A number to identify the order for the Study.
7	Study Description	(0008, 1030)	Institution-generated description.
8	Physician(s) of Record	(0008, 1048)	Names of the physician(s) who are responsible for overall patient care at time of Study
9	Referenced Study Sequence	(0008, 1110)	A sequence that provides reference to a Study SOP Class/Instance pair.
9.1	>Referenced SOP Class UID	(0008, 1150)	Uniquely identifies the referenced SOP Class.
9.2	>Referenced SOP Instance UID	(0008, 1155)	Uniquely identifies the referenced SOP Instance.

General study module is also common to both CT image and RTSTRUCT and hence, we extract this information from an input DICOM CT image for creating the RTSTRUCT file.

(iii) RT Series Module: This module has been created to satisfy the requirements of the standard DICOM Query/Retrieve model while including only those attributes relevant to the identification and selection of radiotherapy objects. Table 4 presents the attributes that we use for creating the RTSTRUCT file.

Table 4: RT series module attributes that we use for writing the RTSRTUCT file.

S.No	Attribute Name	Tag	Attribute Description
1	Modality	(0008, 0060)	Type of equipment that originally acquired the data. It contains enumerated values like RTSTRUCT, RTIMAGE, RTDOSE and RTPLAN.
2	Series Instance User ID	(0020, 000e)	Unique identifier of the series.
3	Series Number	(0020, 0011)	A number that identifies this series.
4	Series Description	(0008, 103e)	User provided description of the series.

This module primarily contains the information of the modality ("RTSTRUCT" in this case) and a series instance user ID that uniquely identifies the series. This ID is created using GDCM [3], a library used by ITK for reading and writing DICOM files.

(iv) General Equipment Module: It specifies the attributes that identify and describe the piece of equipment that produced a series of composite instances. Table 5 presents the attributes that we use for creating the RTSTRUCT file. These attributes are created by the RTSTRUCT-exporter program.

Table 5: General equipment module attributes that we use for writing the RTSRTUCT file.

S.No	Attribute Name	Tag	Attribute Description
1	Manufacturer	(0008, 0070)	Manufacturer of the equipment that produced the composite instances.
2	Station Name	(0008, 1010)	User defined name identifying the machine that produced the composite instances.

(v) Structure Set Module: This module defines a set of areas of significance. Each area can be associated with a frame of reference and zero or more images. Information which can be transferred with each ROI includes geometrical and display parameters, and generation technique. Table 6 presents the attributes that we use for creating the RTSTRUCT file. These attributes are created by the RTSTRUCT-exporter program.

(vi) ROI Contour Module: In general, an ROI can be defined by either a sequence of overlays or a sequence of contours. This module is used to define the ROI as a set of contours. Each ROI contains a sequence of one or more contours, where a contour is either a single point (for a point ROI) or more than one point (representing an open or closed polygon). Table 7 presents the attributes that we use for creating the RTSTRUCT file. These attributes are created by the RTSTRUCT-exporter program.

(vii) RT ROI Observations Module: The RT ROI Observations module specifies the identification and
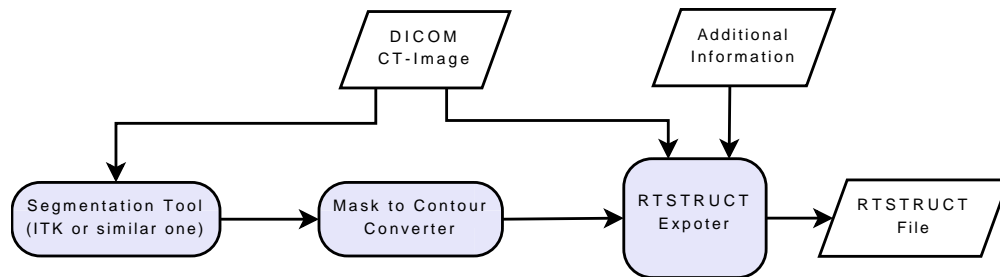
Figure 1: Block diagram illustrating the pipeline for exporting the automated segmentation results to RT-STRUCT.

interpretation of an ROI specified in the Structure Set and ROI Contour modules. Table 8 presents the attributes that we use for creating the RTSTRUCT file. These attributes are created by the RTSTRUCT-exporter program.

(viii) SOP Common Module: The SOP (Service-Object Pair) Common Module is mandatory for all DICOM "Information Object Definitions" (IOD). It defines the attributes which are required for proper functioning and identification of the associated SOP Instances. They do not specify any semantics about the Real-World Object represented by the IOD. Table 9 presents the attributes that we use for creating the RT-STRUCT file. These attributes are created by the RTSTRUCT-exporter program.

In the next Section, we present the implementation details of our RTSTRUCT-exporter.

3   Implementation

Fig. 1 illustrates the pipeline that we use for exporting the automated segmentation results to RTSTRUCT format. It mainly contains three steps: (1) Automated Segmentation, (2)Mask to Contour Conversion and (3) RTSTRUCT-Exporter. Below is a detailed description of these steps:

3.1   Automated Segmentation

The input DICOM CT images are converted into a convenient image format (if required) and an automated segmentation is performed using ITK or similar tools. The output ROIs from this tool should be a mask. There can be multiple masks corresponding to different structures of interest and the current program indeed allows to export multiple masks. It is also possible to export the ROIs obtained on images that are cropped along z-axis; in such cases, the information of starting-slice-number and the number of slices used should be later passed to the RTSTRUCT-exporter module.

The output of this module is passed to the "mask to contour converter".

3.2   Mask to Contour Conversion

We first extract axial slices (in case of a 3-D image) of the mask using ExtractImageFilter of ITK. We then use ContourExtractor2DImageFilter [4] of ITK, for obtaining contours on each of these slices [3]. We

---

[3]Note that as of now, ContourExtractor2DImageFilter is present in "ITK/Code/Review/" directory. Hence, for using this filter, "ITK_USE_REVIEW" option has to be enabled while compiling ITK.

Figure 2: Screen shot of a sample output text file from mask to contour converter.

finally create an output text file containing the information of total number of contours, coordinates of each contour-point along with the corresponding slice number, number of contour points for each contour and type of geometry of each contour (open or closed). We implemented the code for this module in "`mask2contour.cxx`". A screen shot of a sample output text file is shown in Fig. 2. For more details, please refer to the sample file submitted with the code.

While writing the contour data, the values of the attributes: "Image Position" & "Image Orientation" in the original DICOM CT images have to be carefully taken into account. It is because, when a DICOM image is converted to other image formats for performing the segmentation, some of the formats (like raw format) do not represent the absolute coordinates of the voxels. Rather, they represent the offset values and thus result in a translation of the contour coordinates compared to the original DICOM image. Here is a brief description of the above mentioned attributes: The Image Position attribute (tag:(0020, 0032)) specifies the x, y and z coordinates of the upper left hand corner of the image; it is the center of the first voxel transmitted. Image Orientation attribute (tag:(0020, 0037)) specifies the direction cosines of the first row and the first column with respect to the patient. These attributes are provided as a pair. Row value for the x, y and z axes are respectively followed by the column value for the x, y and z axes. The direction of the axes is defined fully by the patient's orientation. The x-axis is increasing to the left hand side of the patient. The y-axis is increasing to the posterior side of the patient. The z-axis is increasing toward the head of the patient. `mask2contour.cxx` program facilitates the specification of translation parameters in x, y and z directions to address this issue.

Another minor point to be noted is that the vertices values given by the ContourExtractor2DImageFilter are in terms of indexes but not in *mm*. Hence, these values are multiplied with their respective voxel spacings in the x, y and z directions in the `mask2contour.cxx` program.

The output of this module is passed to the "RTSTRUCT-Exporter" module.

3.3 RTSTRUCT-Exporter

RTSTRUCT-Exporter is the main contribution of our work. In this subsection, we first present the design details and features of the newly implemented classes. Then we particulary mention the approach that we use for writing sequences because this may be useful in other contexts as well. We then describe the inputs and outputs to the Exporter. Finally, we summarize the list of files that we added/modified, and their potential locations within the ITK directory structure.

Exporting of the contours to RTSTRUCT format requires the implementation of RTSTRUCT-Writer. We implemented the RTSTRUCT-Writer in the "RTSTRUCTIO" class [4]. For creating instances of RTSTRUC-TIO objects using an object factory, "RTSTRUCTIOFactory" class is also implemented. Fig. 3 shows the inheritance diagrams for both RTSTRUCTIO & RTSTRUCTIOFactory classes.
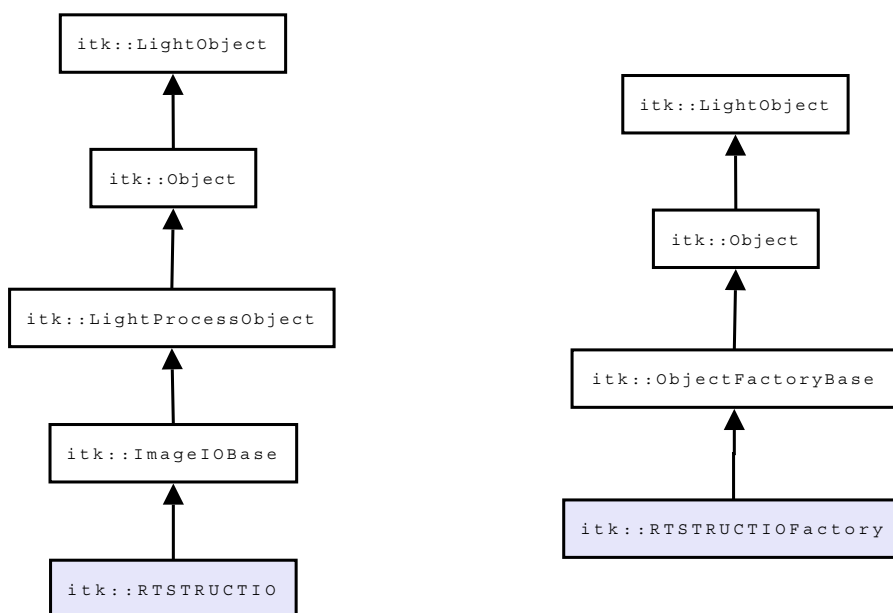

Figure 3: Inheritance diagrams for itk::RTSTRUCTIO & itk::RTSTRUCTIOFactory classes.

Note that these new classes are similar to the existing GDCMImageIO & GDCMImageIOFactory classes. The main difference between GDCMImageIO & RTSTRUCTIO classes is that GDCMImageIO, as the name suggests, is specific to image-based DICOM files whereas the later one is for writing non-image DICOM files. Thus, RTSTRUCTIO class do not contain any image-specific member functions or member variables.

Another important difference between GDCMImageIO & RTSTRUCTIO classes is that GDCMImageIO currently cannot write sequence data whereas RTSTRUCTIO can write sequences. Unlike for DICOM image files, many mandatory attributes of the RTSTRUCT have sequence data. Thus it is essential to handle the sequence data for RTSTRUCT files. Writing sequence data to an RTSTRUCT file is a challenging problem for the following reasons: ITK-GDCM layer does not provide any direct interface for transmitting

---

[4]We have currently implemented only the minimal code that is essential for RTSTRUCT-Exporter. Thus we have implemented only the RTSTRUCT-Writer but not the RTSTRUCT-Reader. However, to keep consistency with the naming convention of other ITK classes, the Writer class is named as RTSTRUCTIO.

the sequence data. Further, a sequence can contain other sequences and also it can have multiple items; but still, a single MetaDataDictionary has to be passed to the RTSTRUCT-Writer.

Here is the approach that we use for writing the sequence data: Each sequence is encapsulated inside a MetaDataDictionary, and this MetaDataDictionary is further encapsulated inside the final MetaDataDictionary which is passed to the RTSTRUCT-Writer [5]. As we mentioned above, each sequence can contain multiple items. So for separating the items within a sequence, they are also encapsulated inside a MetaDataDictionary. But for distinguishing the MetaDataDictionary of sequence from the MetaDataDictionary of item, a predefined string-prefix is used for encapsulating items. For instance, the following prefix is used for distinguishing the encapsulation of an item of DICOM sequence from the encapsulation of the DICOM sequence itself.

```
const std::string ITEM_ENCAPSULATE_STRING("DICOM_ITEM_ENCAPSULATE");
```

While parsing the final MetaDataDictionary in RTSTRUCT-Writer, whenever the MetaDataDictionary entry is another MetaDataDictionary, it is recursively traversed; items within the sequence are identified using the above defined prefix and the sequences are finally written to the RTSTRUCT-file by calling the GDCM library functions in RTSTRUCT-Writer.

We illustrate the above mentioned approach using a simple example of writing Structure Set ROI Sequence (tag: (3006, 0020)), mentioned in Table 6, to RTSTRUCT. A sample code describing the encapsulation this sequence in export2RTSTRUCT.cxx is presented in Listing 1. A sample code describing the recursive subroutine that parses a MetaDataDictionary containing a sequence, and writes to RTSTRUCT file is presented in Listing 2.

The inputs to the RTSTRUCT exporter are:

- An axial slice of the DICOM CT image of the patient (for extracting the information that is common to both CT image & RTSTRUCT, as described in Section 2).

- Output(s) of the Mask to Contour Converter. (Multiple contours can be exported, as described in Section 2)

- Few additional inputs like starting slice number with respect to the original image, total number of slices to be considered, ROI interpreted types and the colors to be assigned to each ROI.

All this information is passed to export2RTSTRUCT executable through a parameter-file. A screen shot of a sample parameter-file is shown in Fig. 4. Please refer to the submitted code for more details.

Finally, we summarize here the list of files that we added/modified, and their potential locations within the ITK directory structure:

- code/IO/ (added)

    - itkRTSTRUCTIO.h
    - itkRTSTRUCTIO.cxx
    - itkRTSTRUCTIOFactory.h
    - itkRTSTRUCTIOFactory.cxx

---

[5]Thanks to Mathieu Malaterre for this suggestion.

Listing 1: Sample code illustrating how sequences are encapsulated in export2RTSTRUCT.cxx.

```cpp
// Values of the attributes
string roiNumber = "1";
string roiName = "External Contour";
string frameOfRefUID =
     "1.2.840.113619.2.55.3.464283444.427.1208496883.210.11425.1";
string roiGenAlgorithm = "ITK-GDCM based algorithm";

// String for distinguishing item vs sequence:
string itemString = ITEM_ENCAPSULATE_STRING + "01";

typedef itk::RTSTRUCTIO          STRUCTOutType;
typedef itk::MetaDataDictionary DictionaryType;

 // Final Dictionary that will be passed to RTSTRUCTIO
STRUCTOutType::Pointer rtstructIO = STRUCTOutType::New();
DictionaryType& dictWriter = rtstructIO->GetMetaDataDictionary();

// Dictionary for encapsulating the sequences.
STRUCTOutType::Pointer structureSetRoi = STRUCTOutType::New();
DictionaryType &structureSetRoiSeq =
               structureSetRoi->GetMetaDataDictionary();

// Dictionary for encapsulating the items within sequences
STRUCTOutType::Pointer rtstructIOItem = STRUCTOutType::New();
DictionaryType &itemDict = rtstructIOItem->GetMetaDataDictionary();

// Encapsulating the attributes of the item...
itk::EncapsulateMetaData<string>(itemDict, "3006|0022", roiNumber);
itk::EncapsulateMetaData<string>(itemDict, "3006|0024", frameOfRefUID);
itk::EncapsulateMetaData<string>(itemDict, "3006|0026", roiName);
itk::EncapsulateMetaData<string>(itemDict, "3006|0036", roiGenAlgorithm);

// Encapsulating item inside the sequence...
itk::EncapsulateMetaData<DictionaryType>
    (structureSetRoiSeq, itemString, itemDict);

// Adding Sequence data to the final dictionary
// that will passed to RTSTRUCTIO...
itk::EncapsulateMetaData<DictionaryType>
    (dictWriter, "3006|0020", structureSetRoiSeq);
```

Listing 2: A subroutine in RTSTRUCTIO.cxx that recursively parses the dictionary containing a sequence.

```cpp
bool RTSTRUCTIO::InsertDICOMSequence(MetaDataDictionary &sub_dict,
                                     const unsigned int itemDepthLevel,
                                     gdcm::SeqEntry     *seqEntry)
{
  gdcm::SQItem *sqItem = NULL;
  itk::MetaDataDictionary::ConstIterator itr = sub_dict.Begin();
  itk::MetaDataDictionary::ConstIterator end = sub_dict.End();
  while(itr != end)
  {
    // Get the key
    const std::string &inside_key = itr->first;

    // If the object inside the sequence is an item, recursively
    // call this subroutine.
    if (ITEM_ENCAPSULATE_STRING ==
          inside_key.substr(0, ITEM_ENCAPSULATE_STRING.length()))
    {
      // Get the dictionary associated with that string...
      itk::MetaDataDictionary actual_sub_dict;
      ExposeMetaData<itk::MetaDataDictionary>(sub_dict, inside_key,
                                              actual_sub_dict);
      InsertDICOMSequence(actual_sub_dict, itemDepthLevel, seqEntry);
    } else {
      if ( itr == sub_dict.Begin() )
        sqItem = new gdcm::SQItem(itemDepthLevel);

      // if the value associated with this key is again a dictionary,
      // recursively call this function to insert a sub-sequence:
      if( typeid(sub_dict).name() == (sub_dict[inside_key].GetPointer())
                                      ->GetMetaDataObjectTypeName())
      {
        itk::MetaDataDictionary inside_sub_dict;
        ExposeMetaData<itk::MetaDataDictionary>(sub_dict,
                                inside_key, inside_sub_dict);

        // if the attached sub-dictinary is a sequence, then.......
        gdcm::SeqEntry *sub_seqEntry = new gdcm::SeqEntry(
          gdcm::Global::GetDicts()->GetDefaultPubDict()->GetEntry(inside_key));
        InsertDICOMSequence(inside_sub_dict, itemDepthLevel+1, sub_seqEntry);
        sqItem->AddEntry(sub_seqEntry);
      } else {// if the value is a string, directly add it to the sequence
        std::string inside_value;
        ExposeMetaData<std::string>(sub_dict, inside_key, inside_value);
        gdcm::ValEntry *inside_valEntry = new gdcm::ValEntry(
         gdcm::Global::GetDicts()-> GetDefaultPubDict()->GetEntry(inside_key));
        inside_valEntry->SetValue(inside_value);

        sqItem->AddEntry(inside_valEntry);
      }
    }
    ++itr;
  }
  if (sqItem != NULL)
    seqEntry->AddSQItem(sqItem, seqEntry->GetNumberOfSQItems());
  return true;
}
```

Figure 4: Screen shot of a sample input parameter-file for RTSTRUCT exporter.

Figure 5: Sagittal, Coronal & Axial views of the 3-D H&N image of the Patient. This is a cropped image containing only the first 83 slices of the original image.

- `Utilities/gdcm/src/` (modified)
  - `gdcmFile.cxx`
  - `gdcmFileHelper.h`
  - `gdcmFileHelper.cxx`
- Other executables (added)
  - `mask2contour.cxx`
  - `export2RTSTRUCT.cxx`

4 Example

The DICOM CT image used in this paper is acquired during routine clinical practice at Divisions of Radiotherapy, Fribourg Hospital (HFR), Switzerland. The image is acquired on GE Medical System (Model: LightSpeedRT16). The size of each slice is 512×512 pixels with a spacing of $1.269531 \times 1.269531\,mm$; the inter-slice distance is $2.5\,mm$. There are 116 slices in total.

Since we are interested only in the first 83 slices of the patient's image, the original DICOM image is cropped in the z-direction to contain only these slice and a new image file (with .mhd extension) is created. The image is then thresholded in selected regions for removing the bed and other immobilization devices. Fig 5 shows the thresholded image [6]. We created separate masks for the external-contour and bones, through simple windowing of the image shown in Fig. 5. These masks are showin in Fig. 6 & Fig. 7.

The contour data of both masks is obtained using `mask2contour`. Note that, as mentioned in Section 3.2, we perform a translation of the contour coordinates based on the Image Position attribute (tag:(0020, 0032)). The contours of the masks along with a slice of the DICOM CT image and and other information is passed to `export2RTSTRUCT`, using the parameter-file submitted along with the code. The correctness of the RTSTRUCT file is verified by inspecting all the tags of the RTSTRUCT file in DicoRView [5], a viewer for DICOM-RT files. DicoRView is a free-tool that provides a very useful tree-view of the attributes. Another free-tool, DICOM_Compare [6], is also used for comparing the contents of different DICOM files. The resultant RTSTRUCT file is also superposed over the original DICOM CT Image in a proprietary software tool, and is shown in Fig. 8. The Exporter is found to correctly create the RTSTRUCT file.

---

[6]Note that the actual images are cropped and resized while displaying in this paper
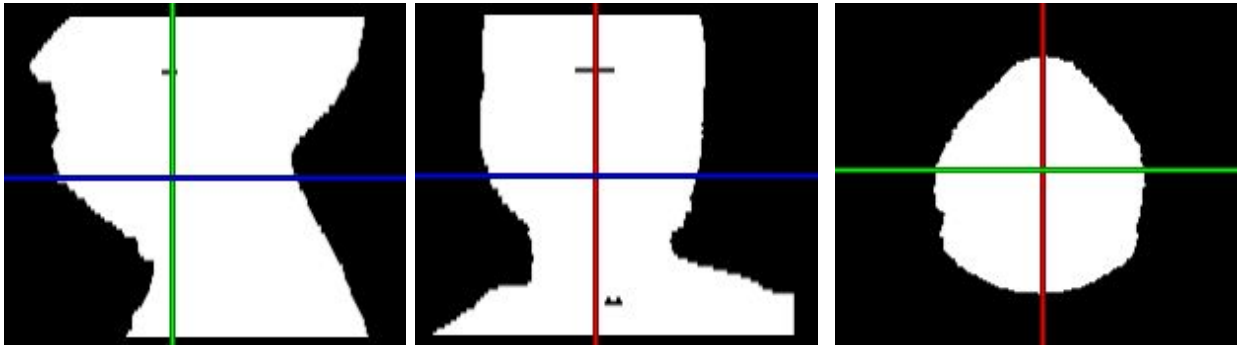
Figure 6: Sagittal, Coronal & Axial views of the mask for external-contour of the 3-D H&N image.

Figure 7: Sagittal, Coronal & Axial views of the mask for bones of the 3-D H&N image.

5 Conclusions & Future Work

An ITK implementation of the RTSTRUCT-Exporter is presented. A summary of the RTSTRUCT format, details of the pipeline used and description of each module in the pipeline are presented. The implementation is validated on a 3-D H&N CT image, by exporting the ROIs of the external-contour and bones to RTSTRUCT format.

Some of the possible extensions of the current work are listed below:

- RTSTRUCT-Exporter is tested only on the DICOM CT images acquired from a GE Medical System (Model: LightSpeedRT16) at Fribourg Hospital, Switzerland. A thorough testing on more images, acquired from various manufacturers and models, will make it more robust.

- At present, only the RTSTRUCT-Writer is implemented but not the RTSTRUCT-Reader. Development of the RTSTRUCT-Reader will be of interest for many general applications.

- RTSTRUCTIO currently inherits ImageIOBase class; however, this does not seem to be the ideal choice since RTSTRUCT is not an image as such.

- There is no direct/simple interface at ITK-layer level, for handling the DICOM sequences. Such interface will be very useful for those who are using ITK for processing DICOM files.

- A complete checker that confirms/warns the user about the presence/correctness of the mandatory attributes of the RTSTRUCT file will be definitely a good contribution to the DICOM/ITK community.
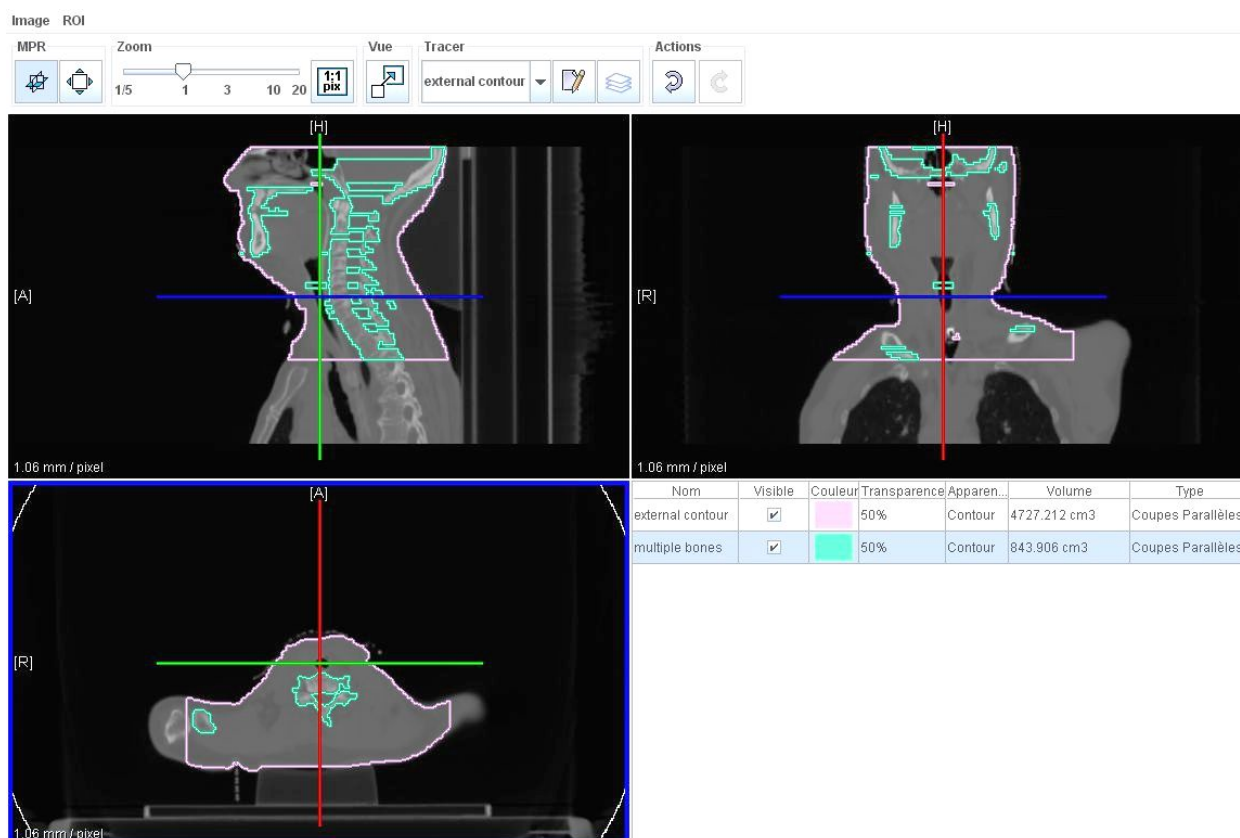
Figure 8: A screen-shot showing the contours of the external-contour and bones in the RTSTRUCT file, superposed over the original DICOM CT image.

Acknowledgments

References

[1] L. Ibanez, W. Schroeder, L. Ng, and J. Cates, *The ITK Software Guide*, 1st ed., Kitware, Inc. ISBN 1-930934-10-6, http://www.itk.org/ItkSoftwareGuide.pdf, 2003. (document)

[2] "DICOM home page." [Online]. Available: http://medical.nema.org/ 2, 2

[3] "GDCM home page." [Online]. Available: http://www.creatis.insa-lyon.fr/Public/Gdcm/ 2

[4] Z. Pincus, "ContourExtractor2DImageFilter: A subpixel-precision image isocontour extraction filter," *Insight Journal*, 2006. [Online]. Available: http://hdl.handle.net/1926/165 3.2

[5] "DICOM-RT viewer." [Online]. Available: http://hscheurig.co.cc/1/dicorview.htm 4

[6] "DICOM-Compare home page." [Online]. Available: http://www.tomovision.com/products/dicom_compare.htm 4

Table 6: Structure set module attributes that we use for writing the RTSRTUCT file.

S.No	Attribute Name	Tag	Attribute Description
1	Structure Set Label	(3006, 0002)	User-defined label for Structure Set.
2	Structure Set Name	(3006, 0004)	User-defined name for Structure Set.
3	Structure Set Date	(3006, 0008)	Date at which Structure Set was last modified.
4	Structure Set Time	(3006, 0009)	Time at which Structure Set was last modified.
5	Referenced frame of Ref. Sequence	(3006, 0010)	Introduces sequence of items describing Frames of Reference in which the ROIs are defined.
5.1	>Frame of Ref. UID	(0020, 0052)	Uniquely identifies Frame of Reference within Structure Set.
5.2	>RT Referenced Study Sequence	(3006, 0012)	Introduces sequence of Studies containing series to be referenced.
5.2.1	>>Referenced SOP Class UID	(0008, 1150)	Uniquely identifies the referenced SOP Class.
5.2.2	>>Referenced SOP Instance UID	(0008, 1155)	Uniquely identifies the referenced SOP Instance.
5.2.3	>>RT Referenced Series Sequence	(3006, 0014)	Introduces sequence of items describing series of images within the referenced study which are used in defining the Structure Set.
5.2.3.1	>>>Series Instance UID	(0020, 000E)	Unique identifier for the series containing the images.
5.2.3.2	>>>Contour Image Sequence	(3006, 0016)	Introduces sequence of items describing images in a given series used in defining the Structure Set.
5.2.3.2.1	>>>>Ref. SOP Class UID	(0008, 1150)	Uniquely identifies the referenced image SOP Class.
5.2.3.2.2	>>>>Ref. SOP Instance UID	(0008, 1155)	Uniquely identifies the referenced image SOP Instance.
6	Structure Set ROI Sequence	(3006, 0020)	Introduces sequence of ROIs for current Structure Set.
6.1	>ROI Number	(3006, 0022)	User-defined name for ROI.
6.2	>Referenced Frame of ref. UID	(3006, 0024)	Uniquely identifies Frame of Reference in which ROI is defined.
6.3	>ROI Name	(3006, 0026)	User-defined name for ROI.
6.4	>ROI Generation Algorithm	(3006, 0036)	Type of algorithm used to generate ROI.

Table 7: ROI Contour module attributes that we use for writing the RTSRTUCT file.

S.No	Attribute Name	Tag	Attribute Description
1	ROI Contour Sequence	(3006, 0039)	Introduces sequence of Contour Sequences defining ROIs.
1.1	>Referenced ROI Number	(3006, 0084)	Uniquely identifies the referenced ROI described in the Structure Set ROI Sequence.
1.2	>ROI Display Color	(3006, 002a)	RGB triplet color representation for ROI.
1.3	>Contour Sequence	(3006, 0040)	Introduces sequence of Contours defining ROI.
1.3.1	>>Contour Image Sequence	(3006, 0016)	Introduces sequence of images containing the contour.
1.3.1.1	>>>Referenced SOP Class UID	(0008, 1150)	Uniquely identifies the referenced image SOP Class of the image containing the Contour.
1.3.1.2	>>>Referenced SOP Instance UID	(0008, 1155)	Uniquely identifies the referenced image SOP Instance of the image containing the Contour.
1.3.2	>>Contour Geometric Type	(3006, 0042)	Geometric type of contour. It contains enumerated Values like: POINT, OPEN_PLANAR, OPEN_NONPLANAR, CLOSED_PLANAR.
1.3.3	>>Number of Contour Points	(3006, 0046)	Number of points (triplets) in Contour Data.
1.3.4	>>Contour Data	(3006, 0050)	Sequence of (x,y,z) triplets defining a contour in the patient based coordinate system.

Table 8: RT ROI observation module attributes that we use for writing the RTSRTUCT file.

S.No	Attribute Name	Tag	Attribute Description
1	RT ROI Observations Sequence	(3006, 0080)	Introduces sequence of observations related to ROIs defined in the ROI Module.
1.1	>Observation Number	(3006, 0082)	Identification number of the Observation.
1.2	>Referenced ROI Number	(3006, 0084)	Uniquely identifies the referenced ROI described in the Structure Set ROI Sequence.
1.3	>RT ROI Interpreted Type	(3006, 00a4)	Type of ROI. It has predefined terms like: PTV, CTV, EXTERNAL, ORGAN etc.
1.4	>ROI Interpreter	(3006, 00a6)	Name of person performing the interpretation.

Table 9: SOP common module attributes that we use for writing the RTSTRUCT file.

S.No	Attribute Name	Tag	Attribute Description
1	SOP Class UID	(0008, 0016)	Uniquely identifies the SOP Class.
2	SOP Instance UID	(0008, 0018)	Uniquely identifies the SOP Instance.
3	Instance Creation Date	(0008, 0012)	Date the SOP Instance was created.
4	Instance Creation Time	(0008, 0013)	Time the SOP Instance was created.
5	Timezone Offset From UTC	(0008, 0201)	Contains the offset from UTC to the time zone for all the attributes present in this SOP Instance.