Data driven Computational Mechanics at EXascale

Data driven Computational Mechanics at EXascale

Work program topic: EuroHPC-01-2019
Type of action: Research and Innovation Action (RIA)

Specification of the UQ aware image segmentation prototype

DELIVERABLE D5.1

Version No 1

http://www.dcomex.eu/

This project has received funding from the European High-Performance Computing Joint Undertaking Joint Undertaking ('the JU'), under Grant Agreement No 956201

DOCUMENT SUMMARY INFORMATION

Project Title	Data driven Computational Mechanics at EXascale
Project Acronym	DCoMEX
Project No:	956201
Call Identifier:	EuroHPC-01-2019
Project Start Date	01/04/2021
Related work package	WP 5
Related task(s)	Task 5.1
Lead Organisation	TUM
Submission date	1/2/2021
Re-submission date	
Dissemination Level	PU

Quality Control:

	Who	Affiliation	Date
Checked by internal reviewer	G. Tetteh	TUM	1/2/2022
Checked by WP Leader			
Checked by Project Coordinator			

Document Change History:

Version	Date	Author (s)	Affiliation	Comment
1.0	1.2.2022	G. Tetteh	TUM	
2.0				

Description

Deliverable 5.1 specifies the prototype of the UQ aware image processing model. It makes algorithms for 3D geometry reconstruction from 3D images available. In this software, 3D image data can be further processed to represent the geometrical domain captured in the 3D images. Standard image processing and deep learning algorithms can be employed for a processing of arbitrary 3D images. Data can be exported to be fed in MSolve in order to generate FEM discretizations and corresponding systems of linear equations to be solved by AI-Solve to process. The image segmentation algorithms can pre-trained for the available applications. Means for an interactive segmentation are available for adjusting the image pre-processing to new tasks, while interfaces to other pre-trained algorithms will be provided.

The DCOMEX image processing tool follows a modular design, and plugins offer well-defined interfaces to established open sources image processing tools (Fig. 1). The plugins offer critical functionality for 2D and 3D image processing, as well as means for export of the generated meshes to MSolve data formats. A description of the general components of the tool is given in Section 1, the plugin manager is described in Section 2, including examples for the integration of different types of external image processing tools, and the algorithms for an MSolve-specific mesh export is described in Section 3.

1. Tool Components

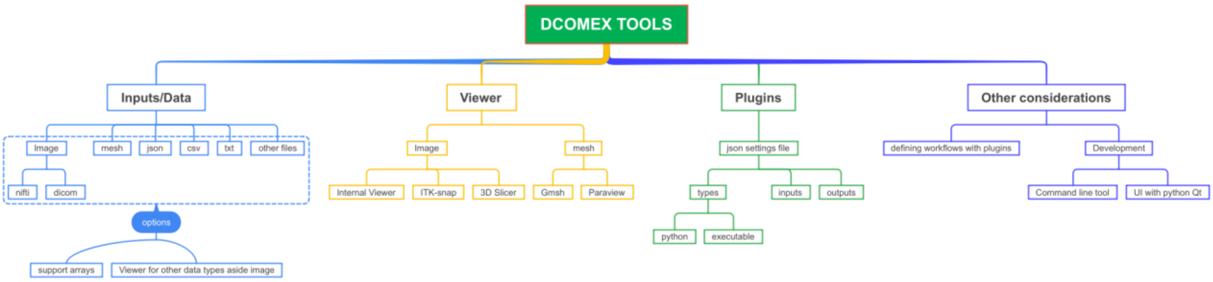
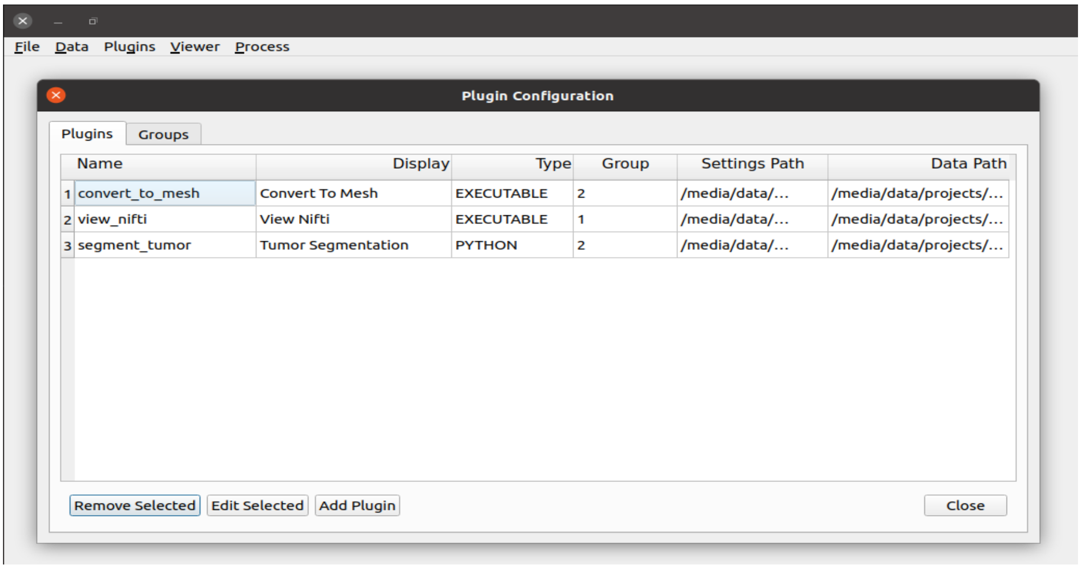


Figure 1: Modular architecture of the DCOMEX image segmentation tool, offering interfaces to image import, visualization, interactive processing, and MSolve-specific meshing.

The main components of the DCOMEX tool are:

- **Workspace Manager:** Data in the tool is organized in workspaces, this makes it possible to save the current workspace (in JSON format) for use at a later time or even on another machine. This can be achieved using a **Data** menu from the GUI. The data store dialog then allows user to upload specific file types (DICOM, NIFTI, MESH, CSV, JSON etc.) and generic files. Output from plugins (Section 2) are automatically added to the datastore.

- **Viewers:** Data visualization (possibly manipulation) happens here. The GUI version of the tool has an in-built image viewer (2D and 3D viewer) and a text file viewer for viewing and editing text file types (JSON, CSV, TXT etc). The idea is not to re-invent the wheel but leverage on existing tools. Hence the viewer support third party viewers like (ITK-Snap, Slicer 3D) through the plugin system.



	Name	Display	Type	Group	Settings Path	Data Path
1	convert_to_mesh	Convert To Mesh	EXECUTABLE	2	/media/data/...	/media/data/projects/...
2	view_nifti	View Nifti	EXECUTABLE	1	/media/data/...	/media/data/projects/...
3	segment_tumor	Tumor Segmentation	PYTHON	2	/media/data/...	/media/data/projects/...

- **Plugins:** The first version of the tool supports two types of plugins – viewers, data processors. A plugin is either written in python or can be a standalone executable. Plugin configurations are uploaded in JSON files and are stored in the tools internal database which are persisted on disk. The plugins are then accessible through the command line tool and also via the GUI (viewer and process menus).

- **Pipelines:** Here users can organize different plugins into a processing pipeline. Example we can build a pipeline which starts with a segmentation plugin and the output from this becomes the input to the image to mesh plugin etc. Once the pipeline is defined, it can then be triggered from the GUI with one-click or through the command line tool.

2. Plugin Manager

The plugin manager is the central element of the DCOMEX tool (Fig 1). A plugin is defined is defined by its name, type, group, working settings, and data path. The **name** is a unique identifier for the plugin and support a display name which is shown in the GUI. The **type** is either a PYTHON script or an EXECUTABLE. The **group** is either one of VIEWER or PROCESSOR. The **settings** file defines how the plugin is executed (details below). The **data path** is a folder with the python module in the case of a PYTHON script plugin.

2. 1. Plugin Settings File (JSON):

The settings required by the plugin is provided in the form of a JSON file and has the following fields.

- **Inputs:** this field is a dictionary of the required input to the plugin, their data type, default values, and whether it is a required input or an optional input.

- **outputs:** this defines the outputs data to expert from the plugin and their data type. This feature is what makes chaining plugins in a single pipeline plugins possible.

- **command/commands:** this is only required for an EXECUTABLE type of plugin and defines the command to be executed by the plugin. This support placeholders which is the substituted at run-time. Multiple conditional commands are supported in the form of a list.

- **callback:** this is applicable in the case of a PYTHON script plugin. It should be assigned the python entry function which is called at run-time. This default to a function call “main”.

- **module:** this is also applicable in the case of a PYTHON script plugin and refers to the python module where the call back function can be found. Defaults to the **data path.**

- **kwargs:** these are any additional params which are sent to the python callback.

2.2. Example 1 – itk-snap as plugin (EXECUTABLE).

External software, such as the widely used and openly available itk-snap viewer and 3D image annotator (www.itk-snap.org) can be integrated into the tool by using a scripted interface. It can be used for interactive segmentation using basic image processing algorithms, such as thresholding or level set segmentation. With a given segmentation, it offers means for an interactive refinement. The example of a script integrating it into the DCOMEX tool reads as follows:

```
{
  "inputs": {
    "raw_input": {
      "display": "Raw Input",
      "type": "image",
      "subtype": ["nifti", “dicom”],
      "description": "The raw image data to be visualized"
    }
  },
  "commands": [
    {
      "command": ["itksnap", "-g", "$i:raw_input"],
      "requires": ["raw_input"]
    },
    {
      "command": ["itksnap"]
    }
  ]
}
```

Here we have two different commands – one for when there is an initial input volume, and one which opens itk-snap without any initial file. Preceding a command entry with “**$i:**” shows that string is a placeholder and will be substituted at run-time.

2.3. Example 2 – Segmentation plugin (PYTHON script)

Other software, for example, pretrained 3D Unet-CNN algorithms can be integrated via the python interface. An example for such a segmentation module reads is the following:

```
{
  "inputs": {
    "raw_input": {
      "display": "Raw Input",
      "type": "image",
      "required": true,
      "subtype": ["nifti", "dicom"],
      "description": "The raw input to be segmented"
    },
  },
  "outputs": {
    "segmentation": {
      "display": "Seg Output",
      "type": "image",
      "subtype": "nifti",
    }
  },
  "callback": "segment_file",
  "python": "python3"
}
```

This plugin calls the function segment_file when executed and will pass it the parameter raw_input, which is the path to the a dicom or nifti file to be segmented. The segment_file function will then return the path to the output segmentation file.

3. Image to Mesh Plugins

After import, segmentation, and interactive visualization and refinement of the 2D or 3D geometry, it needs to be exported to MSolve formats as a mesh (Figure1). To this end, a dedicated meshing and export module is part of the DCOMEX image processing tool. It implements two different type mesh generators, that can either be learned (3.1) or parameterized (3.2).

3.1. Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data

This is a deep learning based method which transforms voxel to mesh through a learned model. This will require training data to adapt the network to different image domains.

- First there is a **training plugin** which handles the training process and returns the trained model as an output.
- Then there is a **prediction plugin** which takes the model and the image data and outputs the mesh.

Reference: Wickramasinghe et al (2020) Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data. Proc MICCAI 2020. https://doi.org/10.1007/978-3-030-59719-1_30

3.2. Standard Approach based on ITK, VTK, and Gmsh open source libraries.

The ITK export does not require any training and makes use of existing open source libraries.

- First the image data is converted to ITK to VTK surface image.
- We then convert the VTK image to a mesh using the Gmsh python API.
- Note: The whole process is handle by a single python plugin. A similar export functionality for interactive meshing, and visualization, and refinement is available in ITK-SNAP (Sec 2.2).

Reference: McCormick M, Liu X, Jomier J, Marion C, Ibanez L. ITK: enabling reproducible research and open science. Front Neuroinform. 2014;8:13. Published 2014 Feb 20. doi:10.3389/fninf.2014.00013