ABSTRACT

Fan-Slicer (https://github.com/UCL/fan-slicer) is a Python package that enables the fast sampling (slicing) of 2D ultrasound-shaped images from a 3D volume. To increase sampling speed, CUDA kernel functions are used in conjunction with the Pycuda package. The main features include functions to generate images from both 3D surface models and 3D volumes. Additionally, the package also allows for the sampling of images from curvilinear (fan shaped planes) and linear (rectangle shaped planes) ultrasound transducers. Potential uses of Fan-slicer include the generation of large datasets of 2D images from 3D volumes and the simulation of intra-operative data among others.
(1) OVERVIEW

INTRODUCTION
Fan-slicer is a package designed for the sampling/simulation of ultrasound shaped planes from a pre-operative scan such as Computed Tomography (CT) or Magnetic Resonance Imaging (MRI). This software has been initially implemented as part of an imaging pipeline to aid the development of ultrasound guidance algorithms for laparoscopic liver surgery [1, 2] and endoscopic interventions [3]. Given a set of Laparoscopic Ultrasound (LUS) images and a pre-operative 3D scan, the resampling of LUS planes in pre-operative space enables both the implementation of image registration pipelines and visualisation of the corresponding results.

MATLAB⁴ and 3D Slicer² have functionalities to perform this sampling operation. However, none of these software tools allow for the fast simulation of smaller planes bounded by the ultrasound fan shape taken at an arbitrary position and orientation in 3D space. Since speed and easy integration of this sampling is a key requirement for registration pipelines in medical imaging applications, we have designed a parallel solution in CUDA that allows for the sampling of multiple planes. Initially implemented in MATLAB with the Parallel Computing Toolbox and CUDA kernels written in C++, the software was later implemented in Python and Pycuda for easier deployment. Currently, this software has enabled research on new registration methods of LUS to CT scans of the liver using Content-based Image Retrieval [1, 2, 4, 5] and the training of General Adversarial Networks (GANs) for the simulation of Ultrasound images from abdominal CT [6].

IMPLEMENTATION AND ARCHITECTURE
Fan-slicer is implemented with Python and CUDA kernels written in C++ that are compiled using Pycuda. A simple overview of the package functionality is described in Figure 1. The project structure is generated from the PythonTemplate of Scikit-Surgery [7]. The package consists of two main classes for the sampling of 2D images – IntensityVolume in intensity_volume.py samples 3D volumes (intensity) and SegmentedVolume in segmented_volume.py samples 3D surfaces (binary). Their implementation is briefly described in Figure 2.

INSTANTIATION AND PRE-ALLOCATION
Upon instantiation, both SegmentedVolume and IntensityVolume allocate volume data in a 3D array. IntensityVolume creates a single array from either a NumPy array, a NifTI file (.nii) or a DICOM file. SegmentedVolume receives a variable number of surfaces in VTK format, performs voxelisation, and then outputs a separate binary 3D array for each of them. As an intermediate step, VTK files are converted to simpler mesh structures described in mesh.py – code can be adapted to other formats as long as this mesh structure is obtained.

To minimise data transfer between CPU and GPU and therefore increase sampling speed, all volume and image data is pre-allocated to the GPU upon instantiation. Besides the 3D data, both classes receive as input a configuration file (.json) with the ultrasound image shape parameterisation and the number of images (an integer) that should be simulated per run. The configuration can be either linear or curvilinear and has variable parameters described in USING.rst. By knowing the configuration and image number to be simulated, the classes pre-allocate a set of fixed size GPU arrays for the slicing task.

IMAGE SAMPLING
Slicing of 2D images is achieved with the function simulate_image for both classes by providing the number of images to slice and an array with a corresponding number

Figure 1 Overview of Fan-slicer package pipelines. Given a virtual ultrasound transducer pose, the package can generate ultrasound shaped images from segmented medical surfaces by voxelising binary volumes or directly from 3D volumetric medical images.
of concatenated 4 × 4 poses composed of rotation and translation. If the number of images is different from the one used in the constructor, the software repeats the pre-allocation step. To generate the input number of images, simulate_image calls a method that uses a sequence of CUDA kernels loaded from cuda_reslicing.py (see Figure 2). Depending on the class and image parameterisation used, a specific combination of kernels listed in Table 1 is used.

For all slicing options, the first kernel computes point clouds from poses (kernels highlighted with (1)). Then, depending on the class (binary or intensity), an interpolation kernel is used (kernels highlighted with (2)). If the configuration is curvilinear, a third kernel must be used to warp the interpolated result into a 2D fan-shaped grid (kernels highlighted with (3)). If the configuration is linear there is no need for a third kernel as the interpolation result is already in linear coordinates.

QUALITY CONTROL
Unit tests in tests/test_pycuda_simulations.py are used to test the image simulation with both intensity and binary models, both with linear and curvilinear shapes. These tests have been checked in Windows and Linux environments. In addition to the tests, simulation_demo.py provides a simple demo on how to simulate images using both intensity and binary models. Therefore, to check if the package is working, a user should:

- Run the unit tests in tests/test_pycuda_simulations.py.
- Run the script simulation_demo.py and check if the plotted image results are the same as the ones stored in the demo_outputs folder.

(2) AVAILABILITY

OPERATING SYSTEM
Minimum versions tested:

2. Windows 10/11 with Windows Subsystem Linux (WSL2).
3. Ubuntu 18.04.5 LTS, with CUDA Toolkit 10.1, gcc 7.5.0 as C++ compiler.

![Figure 2 Overview of the two Fan-slicer image sampling classes. Left shows class instantiation and right shows the image sampling process. Inputs are highlighted in blue, classes in gray, methods in green and dependencies in orange.](image)

<table>
<thead>
<tr>
<th>KERNEL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>transform (1)</td>
<td>Generates point clouds of 3D fan-shaped planes from 4 × 4 poses. Used by slice_volume and intensity_slice_volume.</td>
</tr>
<tr>
<td>linear_transform (1)</td>
<td>Generates point clouds of 3D rectangle-shaped planes from 4 × 4 poses. Used by linear_slice_volume and linear_intensity_slice_volume.</td>
</tr>
<tr>
<td>slice (2)</td>
<td>Uses nearest-neighbour interpolation to map values from a 3D binary array to a set of 3D points. Used by slice_volume and linear_slice_volume.</td>
</tr>
<tr>
<td>weighted_slice (2)</td>
<td>Uses tri-linear interpolation to map values from a 3D array to a set of 3D points. Used by intensity_slice_volume and linear_intensity_slice_volume.</td>
</tr>
<tr>
<td>map_back (3)</td>
<td>Uses nearest-neighbour interpolation to warp a 2D grid of binary values onto a curvilinear/fan-shaped grid. Used by slice_volume.</td>
</tr>
<tr>
<td>intensity_map_back (3)</td>
<td>Uses bi-linear interpolation to warp a 2D grid of intensity values onto a curvilinear/fan-shaped grid. Used by intensity_slice_volume.</td>
</tr>
</tbody>
</table>

Table 1 Description of CUDA kernels in cuda_reslicing.py and functions where these are used. On the left column, a number shows the order in which the kernel is called for simulation.
PROGRAMMING LANGUAGE
Python 3.6, 3.7 and 3.8

ADDITIONAL SYSTEM REQUIREMENTS
Main requirement is a CUDA enabled GPU device with a minimum of 2GB.

DEPENDENCIES
Python packages:
1. Pycuda 2021.1, which requires CUDA Toolkit and a C++ Compiler;
2. NumPy 1.11;
3. VTK;
4. Matplotlib;
5. Scipy;
6. NiBabel;
7. Pydicom.

SOFTWARE LOCATION
Archive
Name: Zenodo
Persistent identifier: 10.5281/zenodo.7387902
Licence: BSD 3-clause
Publisher: João Ramalhinho
Version published: v1.0.1
Date published: 01/12/22

Code repository
Name: GitHub
Persistent identifier: https://github.com/UCL/fan-slicer/
Licence: BSD 3-clause
Date published: 03/03/22

LANGUAGE
Python, C++

(3) REUSE POTENTIAL
This package has the potential to support users that desire either visualise ultrasound shaped sections from 3D volumes or generate large sets of 2D images (e.g. for training neural networks). Additionally, the Pycuda implementations of Fan-slicer are compatible with the CUDA environments of Pytorch and Tensorflow. This means the package allows for the simulation of images during neural network training without storing images in disk.

The package is expected to be continuously supported as authors will maintain the repository and reply to any Github issues.

Contributions to the software could include other ultrasound image shape parameterisations, additional visualisation tools and compatibility with additional image and surface formats.

NOTES
1 https://mathworks.com/.

FUNDING INFORMATIONS
This work is supported in part by the National Institute for Health Research (NIHR) under its Invention for Innovation (i4i) Programme (Grant Reference Number NIHR II-LA-1116-20005), the Wellcome/EPSRC Centre for Interventional and Surgical Sciences (WEISS) [203145Z/16/2] and the EPSRC [EP/T029404/1] grants.

COMPETING INTERESTS
The authors have no competing interests to declare.

AUTHOR AFFILIATIONS
João Ramalhinho © orcid.org/0000-0002-8438-2215
Wellcome/EPSRC Centre for Interventional and Surgical Sciences, University College London, UK

Thomas Dowrick © orcid.org/0000-0002-2712-4447
Wellcome/EPSRC Centre for Interventional and Surgical Sciences, University College London, UK

Ester Bonmati © orcid.org/0000-0001-9217-5438
Wellcome/EPSRC Centre for Interventional and Surgical Sciences, University College London, UK

Matthew J. Clarkson © orcid.org/0000-0002-5565-1252
Wellcome/EPSRC Centre for Interventional and Surgical Sciences, University College London, UK

REFERENCES


