Introduction

Statement

No sensitivities are found therein. The following software does not incorporate data encryption algorithms, it does not incorporate data authentication algorithms, and it cannot be used in massively parallel computing.

General Description

This repository is the implementation of the paper "Deep Learned Generators of Porosity Distributions produced during Metal Additive Manufacturing" (Additive Manufacturing, 2022). The project uses Generative Adversarial Networks and Scattering Transformations to create stochastic realizations of synthetic parts from a dataset of experimental parts. The overall generation process is divided into three subproblems, the generation of individual pores, the generation of the surface boundary, and the recombination of the generated pores with the generated surface boundary to form a completed part.

This dataset used for this work consisted of segmented CT scans of 12 Al-Si-10Mg tensile samples, produced using Laser Powder Bed Fusion. These CT scans consist of individual segmented cross-sections, cut perpendicular to the tensile axis. Each CT scan cross-section consists of the boundary defining the edge of the solid phase of the part, as well as the void spaces that define the pores present in the part.

The overall workflow for this project first analyzes a given experimental CT part sample to extract the individual pore morphologies, surface roughness profiles, and global pore distributions. Next, this information is used to train the Generative Adversarial Network used to generate new pore information, as well as the MST-based microcanonical model necessary to create new surface realizations. During this process, statistical comparisons between the ground truth and synthetic data are made to ensure the key properties of the experimental sample are recreated in the synthetic parts. A hands-on example of this process is demonstrated in the Demo_Part_Generation.ipynb notebook.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.
Based on the description of the code and data, a Derivative Classifier (DC) and a Subject Matter Expert (SME) assert that the code and associated data have no sensitivities and should be deemed Unclassified Unlimited Release (UUR), although they did not physically review every line of code due to its size.

Code Description

Parent directory

Demo_part_generation.ipynb

This file is a jupyter notebook, containing an end-to-end description of the overall generation and analysis process. This notebook contains interactive code examples, written in Python.

utils.py

This file contains helper functions for various operations in the porosity processing sequence, such as normalizing vectors, computing the angle between vectors, and converting to and from polar coordinates. The methods used for these functions are based on the scipy and scikit-image libraries.

images/

This folder contains two images. The first image, data_example.png, provides an example of what the input CT scanned data looks like. The second image, updated_large_schematic.png, provides a visual representation of the overall generation process.

analyze_pore_samples/

analyze_porosity_clean.py

This file is responsible for analyzing the CT scanned porosity samples. This file takes as input segmented .tif images, which it uses to extract the relevant pore information and the surface boundary. During this process, it saves each individual pore as an .hdf5 file, computes the probability matrices necessary, and saves the flattened boundary as an .npy file. The operations for extracting pore regions are carried out based on the Union-Find algorithm implemented in scikit-image, and the properties of the extracted pores are calculated using various image processing techniques defined in the scikit-image regionprops (skimage.measure.regionprops) module.

polar_cartesian_convert.py

Contains helper functions for converting images to polar coordinates.

plotting_utils.py

This file contains helper functions for plotting the figures used in the manuscript. Specifically, there are functions to thicken the size of the axis borders, and create equispaced 3-D plots.

make_surface/

generate_surface.py

This file takes the phase-harmonic variant of the MST, and calls the L-BFGS optimizer to create new realizations of the 2-D profilometry image defining the surface. The code for taking the phase harmonic MST and computing the microcanonical model is taken from an existing GitHub repository, and was not written during a contract with Sandia National Laboratories.

generate_surfaces.py

This file takes the phase-harmonic variant of the MST, and calls the L-BFGS optimizer to create new realizations of the 2-D profilometry image defining the surface. This file has modifications made to the process for saving the generated surfaces to enable bulk generation.

reconstruction/

dataset_test.py

Contains helper functions for saving and loading pores from .hdf5 or .h5 files.

dcgan_test.py

This file contains the code defining architecture of the Generator and the Discriminator networks that comprise a GAN (Generative Adversarial Network).

main_train_pores.py

This file contains the code for training a GAN model on an existing dataset of pores. The model used is defined in dcgan_test.py.

pipeline_clean.py

This file contains the code to generate a new part sample, given an ensemble of generated pores and a set of generated boundary profiles.
This file will save each generated part as a .npz file, containing the pores and boundary images in separate stacks that can be recombined.

pore_generate.py

This file contains the code for loading a saved GAN model and using it to generate an ensemble of pores.

plotting_tools/

create_video.ijm

This file is an imageJ macro to create 3-D animations of the generated porosity distribution for visualization. This file is modified to enable automatic generation from a command line call from within a python script.