Positioning Clamping Variation Pattern for Car Door Reinforcement (Halo) – Case Study

Version 0.0.1, 2019-12-06

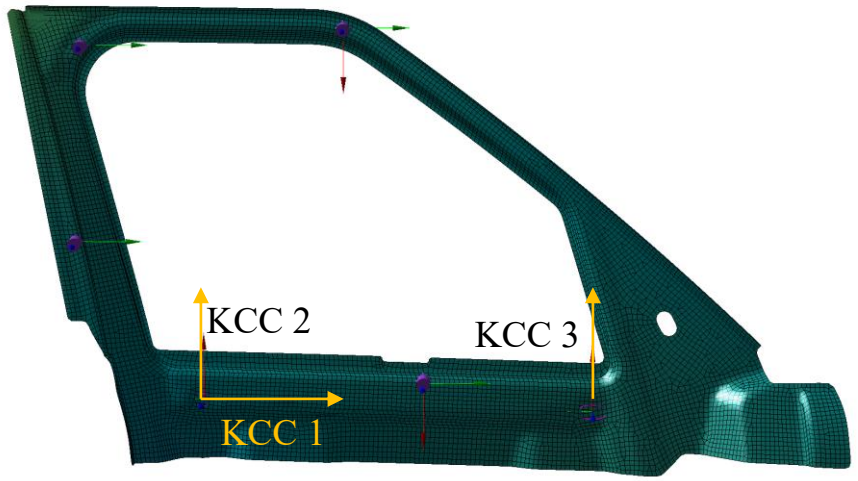
Digital Lifecycle Management (DLM) Research Group

S. Sinha, E. Glorieux, P. Franciosa, D Ceglarek

CASE STUDY - OVERVIEW

Aim

- Study deformation patterns due to locator movement during the positioning & Clamping stage
- Perform Deep Learning using 3D CNN* on the deformation pattern data to estimate process variations causing these deformation patterns



Parameters

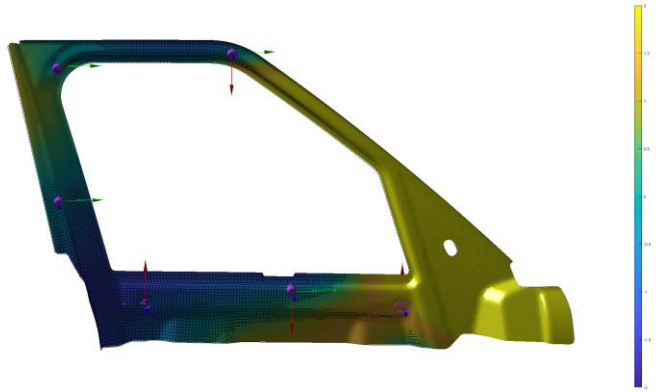
Process Parameter ID**	Description	Unit	Range
KCC 1	Pin hole displacement in x	mm	[-2,2]
KCC 2	Pin hole displacement in z	mm	[-2,2]
KCC 3	Rotation around the pinhole	degree	[-1.5,1.5]

- The part has 8047 mesh nodes

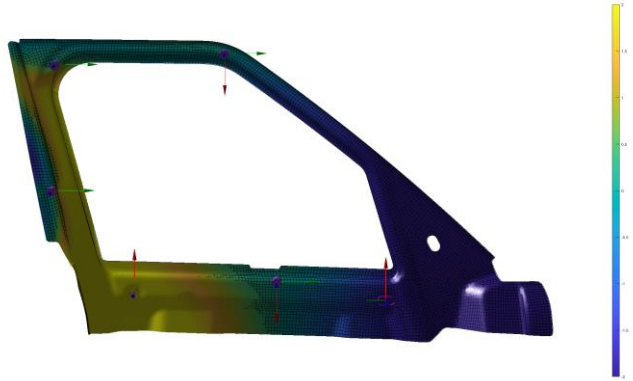
*Sinha, S., Glorieux, E., Franciosa, P., & Ceglarek, D. (2019). 3D convolutional neural networks to estimate assembly process parameters using 3D point-clouds (p. 10). SPIE-Intl Soc Optical Eng. https://doi.org/10.1117/12.2526062

**Process Parameters are termed as Key Control Characteristics (KCCs) in a manufacturing assembly setup

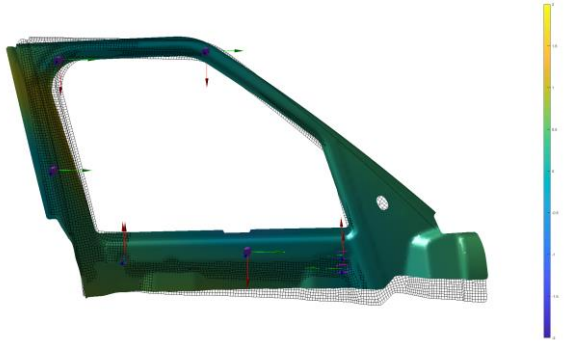
CASE STUDY – DEFORMATION PATTERNS



KCC 1



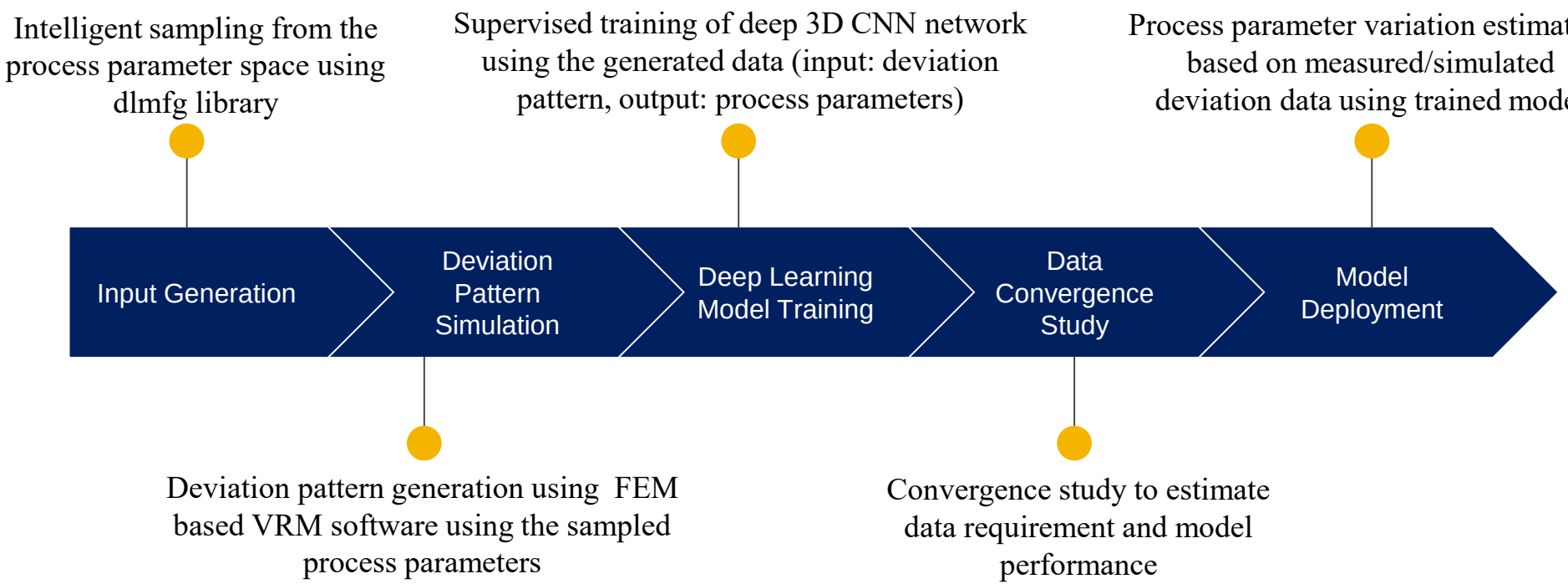
KCC 2



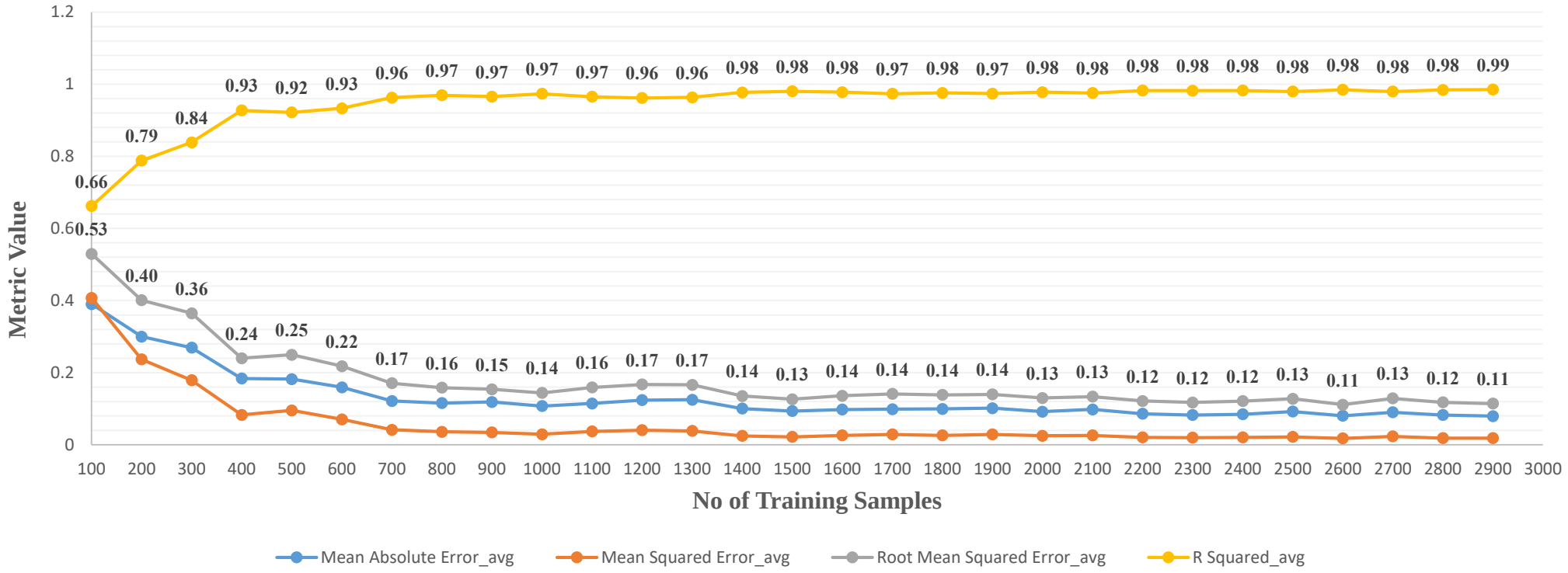
KCC 3

- Deviation patterns generated based on process parameter variations
- Different combination and magnitude of KCC variations give different patterns

CASE STUDY – DATA GENERATION & LEARNING WORKFLOW



CASE STUDY – DATA CONVERGENCE STUDY



- The model converges with 1000 training samples with a root mean square error (RMSE) of 0.14 and R-squared value of 0.97

Html based plot consisting of metrics for each KCC generated using Plotly Python Library embedded below

data_study_plot_test.html

DIGITAL LIFECYCLE MANAGEMENT

CASE STUDY – STEPS FOR MODEL TRAINING AND DEPLOYMENT

1. Download the **dlmfg library** to your python environment *(it is recommended to create a new environment to prevent version mismatch of the prerequisites e.g. TensorFlow, Keras etc)* using Git clone https://github.com/sumitsinha/Deep_Learning_for_Manufacturing or pip install dlmfg *(if you are using git clone you would need to install the requirements by running pip install –r requirements.txt from the root folder)*

2. Run **data_download.py*** located within core that creates the required folder structure and downloads the files within them they consist of :
    1. a folder within datasets with same name of the case study consisting of the model input i.e. node deviations *(Files: output_halo_x.csv, output_halo_y.csv, output_halo_z.csv)*
    2. a folder within active learning with the same name of the case study consisting of the model output i.e. process parameters or KCCs *(Files: initial_samples_halo.csv)*
    3. a file within resources/mapping files consisting of the voxel mapping *(File: voxel_mapping_halo_64.csv)*
    4. a file within resources/nominal_cop consisting of the nominal Cop in oder of node IDs *(File: nominal_cop_halo.csv)*

3. Run **model.train.py*** located within core that trains the model by spitting the train data into train and validation sets and creates an output folder within the trained_models folder with the same name of case consisting of the following file structure:
    1. Logs – model training logs and training metrics (R-squared, RMSE, MAE, MSE)
    2. Model – trained model in Keras trained_model.h5 format
    3. Plots – training loss convergence visualization

4. Run **model_deployment.py*** within core to test the trained model on the test set and obtain accuracy metrics within the logs folder

5. Run **data_study.py*** within model core to train model incrementally by increasing the training data in steps, this obtains accuracy metric at each step by testing the trained model on the test set, this generates html plotly based plots for better visualization and training/testing metrics for each step within the logs folder

** All these files parse the requirement parameters from the case study configuration files within the config folder, each case study has a different set of configuration files which need to be changed in case a new case study within different parameters needs to be trained and deployed*

CASE STUDY – DATA DESCRIPTION

Input Data to the 3D CNN model (Independent variables/Predictor Variables)*:

- x, y, z deviations of 8047 nodes generated as output from the VRM (*Files: output_halo_x.csv, output_halo_y.csv, output_halo_z.csv*)
- These correspond to the output of the VRM simulation software

Output Data of the 3D CNN model (Dependent Variables/Predicted Variables)*:

- Process parameters (KCCs) (*Files: initial_samples_halo.csv*)
- These correspond as input to the VRM simulation software

Mapping Files to map deviation nodes to Voxel locations (64*64*64)

- Voxel index (i, j, k) for each node (x, y, z) *(File: voxel_mapping_halo_64.dat)*

Nominal Cloud of Point with node ID

- Node nominal location in 3D Space *(File: nominal_cop_halo.csv)*

**Split into train and test set, test set contains samples beyond the range of the train set to check for better generalization and prevent use of overfit models*

Thank-you

Digital Lifecycle Management (DLM) Research Group

Please contact Sumit Sinha (email: sumit.sinha.1@warwick.ac.uk) in case of any doubts

S. Sinha, E. Glorieux, P. Franciosa, D Ceglarek