**SigProcOpenPython - User’s Guide**

VDocument/version: *SigProcOpenPython - user's guide v1.0.docx*

**Contents**

Introduction 1

ChangeLog 1

How to run SigProcOpenPython 1

Configuration file (--paramfile) 2

Installation structure 4

Installation Guide 4

How to create models 6

# Introduction

SigProcOpenPython (or SigProcOP) is the Machine Learning part of Safibra OptiGuard product. The main task of SigProcOpenPython is to give the end user to develop the own processing environment within the SigProc infractructure, using python.

data using own envbuild an option to buid his own proc(a) use a Machine Learning model to process input data from OptiGuard sensors to produce (a) value of the vertical offset (“mm“) and value of the absolute vertical offset (“offsetx\_abs).

SigProcOpenPython listens for incoming packets in the form of SigProc, holding the particular mearuements data, collects the packets by timestamp using the internal “collection buffer” and when all measured valus per timestamp are collected, calculates the output values, forms output message and sends to another SigProc for further processing.

# ChangeLog

v1.0

* Initial version

# How to run SigProcOpenPython

After the installation (see below), the system starts automatically, even after the server is restarted.

The status can be verified by:

systemctl status SigProcOpenPython.service

SigProcOpenPython can be also manually run as follows (login as ml user) using the preparatory steps:

sudo systemctl stop SigProcOpenPython.service

cd ~/SigProcOpenPython

conda activate ml

Then can be run either in standard mode

python ogmlrun.py

or in debug mode:

python ogmlrun.py --debug

Additionally, other parameters can be used for ogmlrun.py as follows:

usage: ogmlrun.py [-h] [--paramfile PARAMFILE] [--debug] [--rundir RUNDIR]

[--nolog] [--mode MODE] [--port PORT]

optional arguments:

-h, --help show this help message and exit

--paramfile PARAMFILE

.json file with parameters.

--debug If present, runs in debug mode.

--rundir RUNDIR If present, does cd to the given directory. If not

present, it runs in the directory when the runogml.py

script is present.

--nolog If present, disables logging.

# Configuration file (--paramfile)

Here is an example of the configuration file for SigProcOpenPython with comments:

|  |  |
| --- | --- |
| { |  |
| "name": "SigProcOpenPython", | Name used for logging |
| "description": "configuration for ObjectSense\_01", | Comment - description |
|  |  |
| "maxdblines": 10000, | Maximum number of internal “collection database” messaging buffer where data are collected together by timestamps to form a dataset for mm and offsetx\_abs calculation. If exceeded, some values will be discarded |
| "dbsectoerase": 60, | In seconds. If any measurement coming from input channel is older than dbsectoerase seconds, it is discarded as obsolete |
| "dbmaintainsec": 60, | The interval after which the database messaging buffer is processed to find obsolete measurement data |
| "maxdatabuf": 1000000, | Size of the internal input messages streaming buffer in bytes |
|  |  |
| "listen": { |  |
| "ip": "10.23.18.161", | IP address where SigProcOpenPython listens |
| "port": 5555 | Port where SigProcOpenPython listens |
| }, |  |
|  |  |
| "inputs": [ |  |
| {"name": "ch1", |  |
| "fromdevice": "ProcessGuard\_15", | Measurement 1 device id |
| "fromsensor": "os\_01" | Measurement 1 sensor id |
| }, |  |
| {"name": "ch2", |  |
| "fromdevice": "ProcessGuard\_15", | ... |
| "fromsensor": "os\_02" | ... |
| }, |  |
| {"name": "ch3", |  |
| "fromdevice": "ProcessGuard\_15", | ... |
| "fromsensor": "os\_03" | ... |
| }, |  |
| {"name": "ch4", |  |
| "fromdevice": "ProcessGuard\_15", | ... |
| "fromsensor": "os\_04" | ... |
| }, |  |
| {"name": "ch5", |  |
| "fromdevice": "ProcessGuard\_15", | ... |
| "fromsensor": "os\_05" | ... |
| }, |  |
| {"name": "ch6", |  |
| "fromdevice": "ProcessGuard\_15", | ... |
| "fromsensor": "os\_06" | ... |
| }, |  |
| {"name": "ch7", |  |
| "fromdevice": "ProcessGuard\_15", | ... |
| "fromsensor": "os\_07" | ... |
| }, |  |
| {"name": "ch8", |  |
| "fromdevice": "ProcessGuard\_15", | Measurement 8 device id |
| "fromsensor": "os\_08" | Measurement 8 sensor id |
| } |  |
| ], |  |
|  |  |
| "model\_1": { |  |
| "modelfile": "models/model\_mm.ml", | File with the model for mm |
| "modelname": "SigProcOpenPython model mm" | Model comment |
| }, |  |
|  |  |
| "model\_2": { |  |
| "modelfile": "models/model\_offsetx\_abs.ml", | File with the model for offsetx\_abs |
| "modelname": "SigProcOpenPython model offsetx\_abs" | Model comment |
| }, |  |
|  |  |
| "output\_1": |  |
| {"ip": "10.23.18.73", | IP where SigProcOpenPython is sending output 1 (mm) |
| "port": 5555, | Port where SigProcOpenPython is sending output 1 (mm) |
| "device": "python", | Device id for output 1 (mm) |
| "sensor": "result\_01" | Sensor id for output 1 (mm) |
| }, |  |
|  |  |
| "output\_2": |  |
| {"ip": "10.23.18.73", | IP where SigProcOpenPython is sending output 2 (offsetx\_abs) |
| "port": 5555, | Port where SigProcOpenPython is sending output 1 (offsetx\_abs) |
| "device": "python", | Device id for output 1 (offsetx\_abs) |
| "sensor": "result\_02" | Sensor id for output 1 (offsetx\_abs) |
| } |  |
| } |  |

# Installation structure

/home/ml/SigProcOpenPython/

├── logs

│   └── SigProcOpenPython\_run.log # run log

├── models # directory for models preparation

│   ├── DataRaw.csv # standard input file for models creation

│   ├── model\_mm.ml # standard model file for mm

│   └── model\_offsetx\_abs.ml # standard model file for offsetx\_abs

├── ogmlrun.py # main run code for SigProcOpenPython

├── SigProcOpenPython

│   ├── functionality.py # helper functionality code for SigProcOpenPython

│   ├── helpers # directory with helpers functionality

│   ├── \_\_init\_\_.py

│   └── \_\_pycache\_\_

├── SigProcOpenPython.json # main configuration file

├── README.md

└── run\_ogml.sh # run script called by systemd

# Installation Guide

Installed at Centos 7 server 10.23.18.161

*Installed uner the user ml (sudoer)*

*Installation steps:*

# Download miniconda

cd ~ ; mkdir -p Downloads

cd ~/Downloads

wget <https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh>

\*\*Install Miniconda forgeconda\*\*

cd ~/Downloads

sh Miniconda3-latest-Linux-x86\_64.sh

# read and accept licence (yes)

# install to default /home/ml/miniconda3 location

# confirm "installer" to initialize Miniconda3 (yes)

# relogin as ml

# update miniconda

conda update conda

# create python environment named "ml"

conda create -n ml pip python=3.7 scikit-learn=0.23.1 pandas=1.0.5 matplotlib

# activate python envionment ml

conda activate ml

# download SigProcOpenPython from appropriate git repository

cd ~

git clone [https://github.com/marsal64/SigProcOpenPython.git](https://github.com/marsal64/optiguardml.git)

# regenerate new fresh models

cd ~/SigProcOpenPython/SigProcOpenPython/helpers

python create\_models.py

# verify if everything works (in --debug mode)

cd ~/SigProcOpenPython && python ogmlrun.py --debug

**# add to systemd**

#create new file /lib/systemd/system/SigProcOpenPython.service with contents:

[Unit]

Description=SigProcOpenPython service

After=network-online.target

[Service]

Type=simple

User=ml

WorkingDirectory=/home/ml/SigProcOpenPython

ExecStart=/bin/bash run\_ogml.sh

TimeoutStartSec=0

[Install]

WantedBy=multi-user.target

Alias=SigProcOpenPython.service

**# enable the service, start, verify status**

sudo systemctl daemon-reload

sudo systemctl enable SigProcOpenPython.service

sudo systemctl restart SigProcOpenPython.service

sudo systemctl status SigProcOpenPython.service

**# stop the service**

sudo systemctl stop SigProcOpenPython.service

# How to create models

Warning: don’t do this without exactly knowing what you are doing.

Steps:

* Place file with as /home/ml/SigProcOpenPython/models/DataRaw.csv
* cd /home/ml/SigProcOpenPython/SigProcOpenPython/helpers
* python createmodels.py

Two files are then created in /home/ml/SigProcOpenPython/models/ directory

Possible parameters to be used with createmodels.py:

usage: create\_models.py [-h] [--fromraw] [--rawsubdir RAWSUBDIR]

[--offsubdir OFFSUBDIR] [--chanfname CHANFNAME]

[--startmm STARTMM] [--speed SPEED]

[--testsplit TESTSPLIT] [--randomseed RANDOMSEED]

[--debug] [--inputdata INPUTDATA] [--rundir RUNDIR]

[--modeltype MODELTYPE] [--from\_mm FROM\_MM]

[--to\_mm TO\_MM] [--filemodel\_1 FILEMODEL\_1]

[--filemodel\_2 FILEMODEL\_2] [--mode MODE]

[--port PORT]

OptgiGuartdML create model

optional arguments:

-h, --help show this help message and exit

--fromraw Create RawData.csv from non-mm raw data subdirectories

--rawsubdir RAWSUBDIR

Subdirectory under rundir with raw data subfolders

--offsubdir OFFSUBDIR

Regex to match subdirectory name

--chanfname CHANFNAME

Regex to match channel in file name in subdirectory

--startmm STARTMM Starting value of mm for the first position

--speed SPEED Speed in mm/s

--testsplit TESTSPLIT

Percentage of test, 0.3 means 30 percent of data used

for testing

--randomseed RANDOMSEED

Random seed

--debug If present, does some more detailed profiling and

graphs

--inputdata INPUTDATA

File name of input data

--rundir RUNDIR If present, does cd to the given directory. If not

present, it runs in the directory when the script is

present.

--modeltype MODELTYPE

Model type. Try slower but more precise alternative:

ExtraTreesRegressor

--from\_mm FROM\_MM Drop from mm from input data

--to\_mm TO\_MM Drop to mm from input data

--filemodel\_1 FILEMODEL\_1

Filename for model 1 (mm)

--filemodel\_2 FILEMODEL\_2

Filename for model 2 (offsetx\_abs)

--mode MODE Internal parameter used only by pydev

--port PORT Internal parameter used only by pydev