# VHH Core Package: Automatic Video Analysis Framework (vhh_core)

***Release 1.0.0***

**Daniel Helm**

**Jun 09, 2020**

# CONTENTS

Detecting cinematographic techniques such as Shot Boundary Detection (SBD), Shot Type Classification (STC) and Camera Movements Classification (CMC) is a fundamental task for automatic film archival. Therefore, a software framework is developed to process historical films related to the time of the liberation phase of Nazi concentration camps during the Second World War [Visual History of the Holocaust[1] (VHH)]. This framework is able to detect and classify SBDs (vhh_sbd[3]), STCs (vhh_stc[4]) and CMCs (vhh_cmc[5]) by using deep learning-based as well as optical flow-based approaches [XX][XX][XX]. This documentation is separated into the following sections: In Section *System Overview* and *Process Pipeline* an overview of the code structure as well as the automatic annotation process pipeline is demonstrated. Furthermore, the package structure and the setup and usage description is visualized in Section *Package Structure* and *Setup Instructions*. Finally, this documentation closes with an API description of all classes, modules and their members in Section *API Description*.

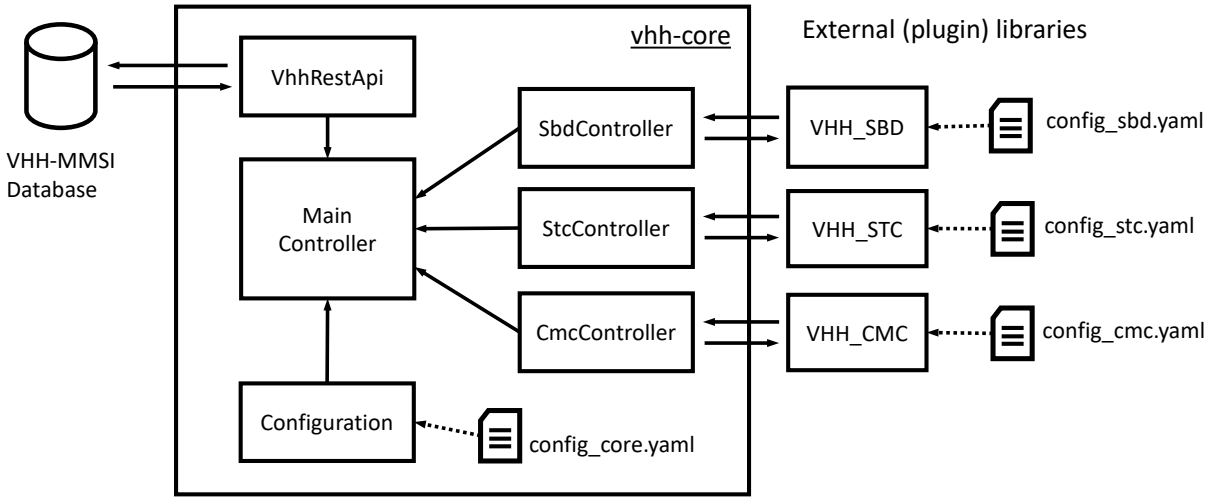
[1] https://www.vhh-project.eu/
[3] https://github.com/dahe-cvl/vhh_sbd
[4] https://github.com/dahe-cvl/vhh_stc
[5] https://github.com/dahe-cvl/vhh_cmc

CHAPTER

# ONE

# SYSTEM OVERVIEW

The software architecture is visualized in the below figure. The vhh_core[2] is divided in multiple classes. The core class is called *MainController* and administers the external plugin modules for automatic shot analysis in historical films: vhh_sbd[3] (SBD), vhh_stc[4] (STC) and vhh_cmc[5] (CMC). Moreover, central functionality such as the communication with the VHH-MMSI system is handled in this class. The class *VhhRestApi* includes methods to communicate with the VHH-MMSI database via RestAPI endpoints. A various number of parameters defined and set in the *config_core.yaml* file is managed by the class *Configuration*. Furthermore, each shot analysis plugin can be configured with separate configuration files stored in the *./config* folder in this repositories.

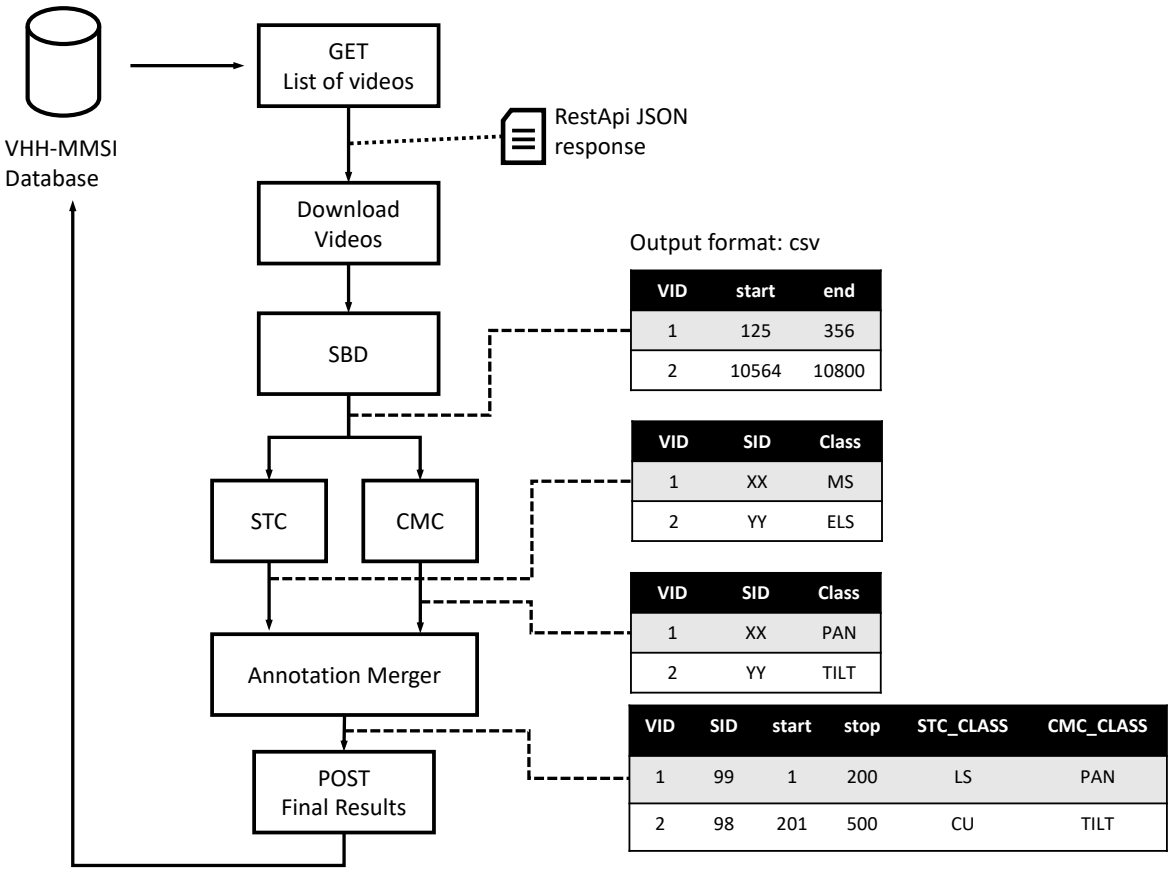



[2] https://github.com/dahe-cvl/vhh_core

CHAPTER

TWO

# PROCESS PIPELINE




| VID | start | end |
|---|---|---|
| 1 | 125 | 356 |
| 2 | 10564 | 10800 |

| VID | SID | Class |
|---|---|---|
| 1 | XX | MS |
| 2 | YY | ELS |

| VID | SID | Class |
|---|---|---|
| 1 | XX | PAN |
| 2 | YY | TILT |

| VID | SID | start | stop | STC_CLASS | CMC_CLASS |
|---|---|---|---|---|---|
| 1 | 99 | 1 | 200 | LS | PAN |
| 2 | 98 | 201 | 500 | CU | TILT |

CHAPTER

THREE

# PACKAGE OVERVIEW

The following list gives an overview of the folder structure of this python repository:

*name of repository*: vhh_core

- **ApiSphinxDocumentation/**: includes all files to generate the documentation as well as the created documentations (html, pdf). The results are stored in the build folder (e.g. xx/build/latex/vhh_core.pdf)
- **config/**: This folder includes the required configuration file. For each plugin (sbd, stc, cmc, ...) there is one subdirectory holding the corresponding configuration file.
- **Demo/**: This folder includes a all scripts to run this application. Furthermore, scripts to setup the folder structure (video and results storage) as well as to setup the environment for running this application.
- **Develop/**: This folder includes additional scripts used during developing phase of this application. (e.g. the document_builder script to generate the package documentation).
- **README.md**: This file gives a brief description of this repository (e.g. link to this documentation)
- **requirements.txt**: this file holds all python dependencies and is needed to install the package in your own virtual environment
- **setup.py**: this script is needed to install the stc package in your own virtual environment

CHAPTER

FOUR

# SETUP INSTRUCTIONS

This package includes a setup.py script and a requirements.txt file which are needed to install this package for custom applications. The following instructions have to be done to used this library in your own application:

**Requirements:**

- Ubuntu 18.04 LTS
- CUDA 10.1 + cuDNN
- python version 3.6.x

**Create a virtual environment:**

- create a folder to a specified path (e.g. /xxx/vhh_core/)
- python3 -m venv /xxx/vhh_core/

**Activate the environment:**

- source /xxx/vhh_core/bin/activate

**Checkout vhh_core repository to a specified folder:**

- git clone https://github.com/dahe-cvl/vhh_core

**Install the stc package and all dependencies:**

- change to the root directory of the repository (includes setup.py)
- python setup.py install

---

**Note:** You can check the success of the installation by using the commend *pip list*. This command should give you a list with all installed python packages and it should include *vhh_core*

---

---

**Note:** Currently there is an issue in the *setup.py* script. Therefore the pytorch libraries have to be installed manually by running the following command: *pip install torch==1.5.0+cu101 torchvision==0.6.0+cu101 -f https://download.pytorch.org/whl/torch_stable.html*

---

CHAPTER

FIVE

# PARAMETER DESCRIPTION

DEBUG_FLAG This parameter is used to activate or deactivate the debug mode.

PEM_PATH This parameter specifies the path tp the certificate file needed to provide vhh-mmsi api access.

ROOT_URL This parameter specifies the root url for the RestApi endpoints.

VIDEO_DOWNLOAD_PATH This parameter specifies the video download path.

CLEANUP_FLAG This parameter is used to activate/deactivate the clean-up mode. The clean-up mode deletes all downloaded videos and the corresponding generated results.

RESULTS_ROOT_DIR This parameter specifies the results path for each individual plugin.

SBD_CONFIG_FILE This parameter specifies the configuration path (.yaml) for the shot boundary detection plugin module

STC_CONFIG_FILE This parameter specifies the configuration path (.yaml) for the shot type classification plugin module

CMC_CONFIG_FILE This parameter specifies the configuration path (.yaml) for the camera movement classification module

CHAPTER

SIX

# API DESCRIPTION

This section gives an overview of all classes and modules in *vhh_core* as well as a brief configuration parameter description.

## 6.1 Configuration class

**class** Configuration.**Configuration**(*config_file: str*)

Bases: object

This class is needed to read the configuration parameters specified in the configuration.yaml file. The instance of the class is holding all parameters during runtime.

---

**Note:** e.g. ./config/CORE/config.yaml

the yaml file is separated in multiple sections config['Development'] config['Security'] config['VhhCore'] config['ApiEndpoints'] config['PluginConfigs']

whereas each section should hold related and meaningful parameters.

---

**loadConfig**()

Method to load configurables from the specified configuration file

## 6.2 Sbd class

**class** Sbd.**Sbd**(*config=None*)

Bases: object

This class includes the interfaces and methods to use the plugin package SBD.

**run**(*video_instance_list=None*)

This method is used to run the shot boundary detection task

**Parameters** **video_instance_list** – parameter must hold a list of video objects (Class-type: Video)

## 6.3 Stc class

**class** Stc.**Stc**(*config=None*)
Bases: object

This class includes the interfaces and methods to use the plugin package STC.

**run**()
This method is used to run the shot type classification task.

## 6.4 Video class

**class** Video.**Video**(*config=None*)
Bases: object

This class represents a video object.

**cleanup**()
This method is used to cleanup all data related to the corresponding video ID. It deletes the generated results of the sbd, stc and cmc plugin as well as the downloaded video file.

**create_video**(*vid*, *originalFileName*, *url*, *download_path*)
This method is used to fill all properties of a video.

**Parameters**

- **vid** – This parameter must hold a valid video id.
- **originalFileName** – This parameter must hold the original filename of a video
- **url** – This parameter must hold the download url of the video.
- **download_path** – This parameter must hold the download path in the local storage.

**download**(*rest_api_instance=None*)
This method is used to download the video into the local storage path.

**Parameters** **rest_api_instance** – This parameter must hold a valid VhhRestApi object.

**Returns** This method returns the download status (true ... successfully downloaded OR false ... download failed).

**is_downloaded**()
This method is used to check if a video is already downloaded.

**Returns** This method returns a boolean flag (true ... video already downloaded OR false ... video does not exist).

**printInfo**()
This method summarizes all properties of this object and print it to the console.

## 6.5 VhhRestApi class

**class** VhhRestApi.**VhhRestApi**(*config=None*)
Bases: object

This class includes the interfaces and methods to use the vhh restAPI interfaces provided by MaxRecall.

**downloadVideo**(*url*, *file_name*, *video_format*)
This method is used to download a video from the Vhh-MMSI system.

**Parameters**

- **url** – this parameter must hold a valid restApi endpoint.
- **file_name** – This parameter represents the filename on the local storage.
- **video_format** – This parameter must hold a valid video format extension (e.g. m4v)

**Returns** This method returns a boolean flag which includes the state of the download process (true ... sucessfully finished OR false ... downlaod failed)

**getAutomaticResults**(*vid*)
This method is used to get all automatic generated results from the VhhMMSI system.

**Parameters** **vid** – This parameter must hold a valid video identifier.

**Returns** THis method returns the results (payload) as json format.

**getListofVideos**()
This method is used to get a list of all available videos in the VHH-MMSI system.

**Returns** This method returns a list of video objects (Class-type: Video) which holds all video specific meta-data.

**getRequest**(*url*)
This method is used to send a get request to the Vhh-MMSI system.

**Parameters** **url** – this parameter must hold a valid restApi endpoint.

**Returns** This method returns the original response including header as well as payload.

**postAutomaticResults**(*vid*, *results_np*)
This method is used to post the automatic generated results to the VhhMMSI system.

**Parameters**

- **vid** – This parameter must hold a valid video identifier.
- **results_np** – This parameter must hold a numpy array including the automatic generated results.

**postRequest**(*url*, *data_dict*)
This method is used to send a post request to the Vhh-MMSI system.

**Parameters**

- **url** – this parameter must hold a valid restApi endpoint.
- **data_dict** – this parameter must hold a valid list of dictionaries with the specified fields (see RestApi documentation).

**Returns** This method returns the original response including header as well as payload

## 6.6 MainController class

CHAPTER

SEVEN

# INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

EIGHT

# REFERENCES

# INDEX