VHH Plugin Package: Camera Movements Classification (vhh_cmc)

Release 1.0.0

Daniel Helm

Jun 08, 2020

CONTENTS

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

name of repository: vhh_cmc

- **ApiSphinxDocumentation/**: includes all files to generate the documentation as well as the created documentations (html, pdf)
- **config/**: this folder includes the required configuration file
- **cmc/**: this folder represents the shot-type-classification module and builds the main part of this repository
- **Demo/**: this folder includes a demo script to demonstrate how the package have to be used in customized applications
- **Develop/: includes scripts to generate the sphinx documentation. Furthermore, a script is included to run a** process to evaluate the implemented approach on a specified dataset.
- **README.md**: this file gives a brief description of this repository (e.g. link to this documentation)
- **requirements.txt**: this file holds all python lib dependencies and is needed to install the package in your own virtual environment
- **setup.py**: this script is needed to install the cmc package in your own virtual environment

CHAPTER

ONE

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 use this library in your own application:

Requirements:

- Ubuntu 18.04 LTS
- python version 3.6.x

Create a virtual environment:

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

Activate the environment:

- source /xxx/vhh_cmc/bin/activate

Checkout vhh_cmc repository to a specified folder:

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

Install the cmc package and all dependencies:

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

Setup environment variables:

- source /data/dhelm/python_virtenv/vhh_sbd_env/bin/activate
- export CUDA_VISIBLE_DEVICES=1
- export PYTHONPATH=$PYTHONPATH:/XXX/vhh_cmc/:/XXX/vhh_cmc/Develop/:/XXX/vhh_cmc/Demo/

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_cmc*.

Run demo script

- change to root directory of the repository
- python Demo/vhh_cmc_run_on_single_video.py

CHAPTER

TWO

PARAMETER DESCRIPTION

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

SBD_RESULTS_PATH This parameter is used to specify a SBD results file for debugging mode.

SAVE_DEBUG_PKG This parameter is used to save a debug package (e.g. including some visualizations, ... - not available yet).

CONVERT2GRAY_FLAG This flag is used to convert a input frame into a grayscale frame (0... deactivate, 1 ... activate).

CENTER_CROP_FLAG This flag is used to center crop a input frame (0... deactivate, 1 ... activate).

DOWNSCALE_FLAG This flag is used to scale a input frame into the specified dimension (0... deactivate, 1 ... activate).

RESIZE_DIM This flag is used to to specify the resize dimension. (only usable if DOWNSCALE_FLAG is active).

SENSITIVITY This parameter is used to specify the number of consecutive frames which are needed to register a camera movement.

SPECIFICITY This parameter is used to specify the number of outliers (miss detections).

BORDER This parameter is used to specify the frame border inside which random features are created

NUMBER_OF_FEATURES This parameter is used to specify the number of features number of features to be tracked for optical flow

ANGLE_DIFF_LIMIT This parameter is used to specify the difference limit to most common angle such that still considered as background movement

MODE NORMAL_MODE = 0 DEBUG_MODE = 1 SAVE_MODE = 2 DEBUG_AND_SAVE_MODE = 3

CLASS_NAMES This parameter is used to specify the class names.

SAVE_RAW_RESULTS This parameter is used to save raw results (e.g. debug visualizations).

PATH_RAW_RESULTS This parameter is used to specify the path for saving the raw results.

PREFIX_RAW_RESULTS This parameter is used to specify the prefix for the results file.

POSTFIX_RAW_RESULTS This parameter is used to specify the postfix for the results file.

SAVE_FINAL_RESULTS This parameter is used to save final results (e.g. csv list).

PATH_FINAL_RESULTS This parameter is used to specify the path for saving the final results.

PREFIX_FINAL_RESULTS This parameter is used to specify the prefix for the results file.

POSTFIX_FINAL_RESULTS This parameter is used to specify the postfix for the results file.

PATH_VIDEOS This parameter is used to specify the path to the videos.

SAVE_EVAL_RESULTS This parameter is used to save evaluation results (e.g. visualizations, . . .).

PATH_RAW_RESULTS This parameter is used the raw results path.

PATH_EVAL_RESULTS This parameter is used to specify the path to store the evaluation results path.

PATH_GT_ANNOTATIONS This parameter is used to groundtruth annotations used for evaluation.

PATH_EVAL_DATASET This parameter is used to specify the path to the dataset used for the evaluation.

CHAPTER

THREE

API DESCRIPTION

This section gives an overview of all classes and modules in *cmc* as well as an code description.

3.1 Configuration class

class cmc.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/config_vhh_test.yaml

the yaml file is separated in multiple sections config['Development'] config['PreProcessing'] config['CmcCore'] config['Evaluation']

whereas each section should hold related and meaningful parameters.

---

loadConfig()
Method to load configurables from the specified configuration file

3.2 CMC class

class cmc.CMC.**CMC**(*config_file: str*)
Bases: object

Main class of camera movements classification (cmc) package.

exportCmcResults(*fName*, *cmc_results_np: numpy.ndarray*)
Method to export cmc results as csv file.

Parameters

- **fName** – [required] name of result file.
- **cmc_results_np** – numpy array holding the camera movements classification predictions for each shot of a movie.

loadSbdResults(*sbd_results_path*)
Method for loading shot boundary detection results as numpy array

Note: Only used in debug_mode.

Parameters **sbd_results_path** – [required] path to results file of shot boundary detection module (vhh_sbd)

Returns numpy array holding list of detected shots.

runOnSingleVideo(*shots_per_vid_np=None*, *max_recall_id=- 1*)
Method to run cmc classification on specified video.

Parameters

- **shots_per_vid_np** – [required] numpy array representing all detected shots in a video (e.g. sid | movie_name | start | end)
- **max_recall_id** – [required] integer value holding unique video id from VHH MMSI system

3.3 OpticalFlow class

class cmc.OpticalFlow.**OpticalFlow**(*video_frames=None*, *fPath=''*, *debug_path=''*, *sf=0*, *ef=1*, *mode=0*, *pan_classifier=<cmc.OpticalFlow.AngleClassifier object>*, *tilt_classifier=<cmc.OpticalFlow.AngleClassifier object>*, *sensitivity=20*, *specificity=3*, *border=50*, *number_of_features=100*, *angle_diff_limit=20*, *config=None*)
Bases: object

This class is used for optical flow calculation.

__init__(*video_frames=None*, *fPath=''*, *debug_path=''*, *sf=0*, *ef=1*, *mode=0*, *pan_classifier=<cmc.OpticalFlow.AngleClassifier object>*, *tilt_classifier=<cmc.OpticalFlow.AngleClassifier object>*, *sensitivity=20*, *specificity=3*, *border=50*, *number_of_features=100*, *angle_diff_limit=20*, *config=None*)
Constructor.

Parameters

- **video_frames** – This parameter holds a valid numpy array representing a range of frames (e.g. NxWxHxchannels).
- **fPath** – This parameter holds a valid path consisting of the absolute path and the correct video name.
- **debug_path** – This parameter specifies a valid path to store results in debug mode.
- **sf** – This parameter represents the starting frame index.
- **ef** – This parameter represents the ending frame index.
- **mode** – This parameter represents runtime mode (e.g. DEBUG_MODE=1 or SAVE_MODE=2).
- **pan_classifier** – This parameter holds a valid object of class type AngleClassifier.
- **tilt_classifier** – This parameter holds a valid object of class type AngleClassifier.
- **sensitivity** – This parameter is used to configure the optical flow algorithm.
- **specificity** – This parameter is used to configure the optical flow algorithm.

- **border** – This parameter is used to configure the optical flow algorithm.
- **number_of_features** – This parameter is used to configure the optical flow algorithm.
- **angle_diff_limit** – This parameter is used to configure the optical flow algorithm.
- **config** –

optical_flow(*prev_frame*, *prev_feat*, *curr_frame*)
This method is used to calculate the optical flow between two consecutive frames.

Parameters

- **prev_frame** – This parameter must hold a valid numpy frame (previous).
- **prev_feat** – This parameter must hold valid features of the previous frame.
- **curr_frame** – This parameter must hold a valid numpy frame (current).

Returns This method returns two arrays including the features of the previous frame as well as of the current frame.

run()
This method is used to run the optical flow calculation process.

Returns This method returns a separate list for each movement class and holds the predicted frame ranges of both.

run_eval()
This method is used to run the optical flow calculation to evaluate specified videos.

run_manual_evaluation()
This method is used to run optical flow process in DEBUG mode. A valid X-Server is needed to visualize the frame player.

3.4 PreProcessing class

class cmc.PreProcessing.**PreProcessing**(*config_instance:* cmc.Configuration.Configuration)
Bases: object

This class is used to pre-process frames.

applyTransformOnImg(*image: numpy.ndarray*) → numpy.ndarray
This method is used to apply the configured pre-processing methods on a numpy frame.

Parameters **image** – This parameter must hold a valid numpy image (WxHxC).

Returns This methods returns the preprocessed numpy image.

convertRGB2Gray(*img: numpy.ndarray*)
This method is used to convert a RBG numpy image to a grayscale image.

Parameters **img** – This parameter must hold a valid numpy image.

Returns This method returns a grayscale image (WxHx1).

crop(*img: numpy.ndarray*, *dim: tuple*)
This method is used to crop a specified region of interest from a given image.

Parameters

- **img** – This parameter must hold a valid numpy image.

- **dim** – This parameter must hold a valid tuple including the crop dimensions.

Returns This method returns the cropped image.

resize(*img: numpy.ndarray*, *dim: tuple*)
This method is used to resize a image.

Parameters

- **img** – This parameter must hold a valid numpy image.
- **dim** – This parameter must hold a valid tuple including the resize dimensions.

Returns This method returns the resized image.

3.5 Evaluation class

class cmc.Evaluation.**Evaluation**(*config_instance:* cmc.Configuration.Configuration)
Bases: object

Evaluation class includes all methods to evaluate the implemented algorithms.

calculate_metrics(*y_score*, *y_test*)
This method is used to calculate the metrics: precision, recall, f1score. Furthermore, the confusion matrix is generated and stored as figure on a specified location.

Parameters

- **y_score** – This parameter must hold a valid numpy array with the class prediction per shot .
- **y_test** – This parameter must hold a valid numpy array with the groundtruth labels per shot.

load_dataset()
This method is used to load the dataset used to evaluate the algorithm. The dataset must have the following structure: dataset_root_dir/training_data/ dataset_root_dir/training_data/tilt/ dataset_root_dir/training_data/pan/ dataset_root_dir/training_data/annotation/xxx.flist

plot_confusion_matrix(*cm=None*, *target_names=[]*, *title='Confusion matrix'*, *cmap=None*, *normalize=True*, *path=''*)
given a sklearn confusion matrix (cm), make a nice plot

cm: confusion matrix from sklearn.metrics.confusion_matrix

target_names: given classification classes such as [0, 1, 2] the class names, for example: ['high', 'medium', 'low']

title: the text to display at the top of the matrix

cmap: the gradient of the values displayed from matplotlib.pyplot.cm see http://matplotlib.org/examples/color/colormaps_reference.html plt.get_cmap('jet') or plt.cm.Blues

normalize: If False, plot the raw numbers If True, plot the proportions

plot_confusion_matrix(cm = cm, # confusion matrix created by

sklearn.metrics.confusion_matrix

normalize = True, # show proportions target_names = y_labels_vals, # list of names of the classes title = best_estimator_name) # title of graph

http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html :param cm: :param target_names: :param title: :param cmap: :param normalize: :param path:

run_evaluation()
This method is used to start and run the evaluation process.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

4.1 References

INDEX

Symbols

A

C

E

L

O

P

R