Flamingo Documentation

Release 1.0.1

Bas Hoonhout

July 11, 2017

Contents

The Flamingo toolbox is an open-source toolbox for image segmentation, classification and rectification. It is developed by the Department of Hydraulic Engineering of Delft University of Technology for coastal image analysis. The toolbox is built around the *scikit-image*, *scikit-learn*, *OpenCV* and *pystruct* toolboxes.

Flamingo is developed and maintained by:

Bas Hoonhout <b.m.hoonhout@tudelft.nl>
Max Radermacher <m.radermacher@tudelft.nl>

The toolbox can be found at http://github.com/openearth/flamingo.

CHAPTER 1

Contents

Rectification

This module provides functions to project an image onto a real-world coordinate system using ground control points. The module is largely based on the *OpenCV Camera Calibration and 3D Reconstruction* workflow and works nicely together with the *argus2* toolbox for coastal image analysis.

A typical workflow consists of determining ground control points by measuring the real-world coordinates of object visible in the image and the image coordinates of these very same objects. Also the camera matrix and lens distortion parameters should be determined.

Subsequently, a homography can be determined using the `rectification.rectification.find_homography()` function and a projection of the image can be plotted using the accompanying `rectification.plot` module.

See also:

http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

Rectification

Visualization

Segmentation

This module provides functions to segmentate an image into superpixels. It is largely based on the *scikit-image* toolbox. Apart from the regular segmentation functions it provides postprocessing functions to ensure connected segments in a regular grid. It also provides various visualization tools for segmented images.

See also:

http://scikit-image.org/docs/0.10.x/api/skimage.segmentation.html

Superpixels

Postprocessing

Visualization

Classification

This module provides functions to train image classification models, like Logistic Regressors and Conditional Random Fields. It provides functions for feature extraction that are largely based on the *scikit-image* toolbox and it provides functions for model training and optimization that are largely based on the *pystruct* and *scikit-learn* toolbox.

See also:

http://scikit-image.org/docs/0.10.x/api/skimage.feature.html

See also:

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model

See also:

https://pystruct.github.io/references.html

Models

Features

Blocks

Scale invariant features

Normalizing features

Relative location prior

Channels

Test

Visualization

Utils

Calibration

In development.

CHAPTER 2

Command-line tools

Several command-line functions are supplied with the toolbox for batch processing of large datasets. Each command-line function serves a specific part of the image analysis. See for more information the *–help* option of each command.

rectify-images

classify-images

calibrate-camera

CHAPTER 3

File system

The toolbox uses a file system structure for the analysis of datasets. The `filesys` module takes care of any reading and writing of files in this file structure. Each dataset is stored in a single directory and can consist out of the following file types:

Image files Any image file recognized by the system

Cropped image files Names start with *cropped_*. A non-cropped version of the image file should exist.

Export files Pickle files with data concerning an image. Each export file name has the following format: *<image_name>.<key>.pkl*. A special type of export file is the feature file. Not all features are written to a single export file, but they are subdivided into multiple export files depending on the feature block they belong to. The block name is added to the export file, just before the file extension.

Log files Pickle files with data concerning the entire dataset. Log file names can have any name.

Model files Pickle files with a trained model. Each model file is accompanied by a meta file. Each model file name has the following format: *model_<model_type>_<dataset>_I<nr_of_images>_B<nr_of_blocks>_<timestamp>.pkl*. The corresponding meta file has *meta* added to the name, just before the file extension.

CHAPTER 4

Configuration

Only the very basic options of the toolbox are exposed through the command-line functions. For the full extent of options a configuration file is used. This configuration file is parsed by the `config` module. The module also supplies wrappers for the automated updating of a function call based on the configuration file used.

`config.`**`CLASSIFICATION_DEFAULTS`** **= {'channels': {'enabled': True, 'methods': ['gabor', 'gaussian', 'sobel'], 'methods_p**

Configuration constants for classification toolbox

`config.`**`get_function_args`**(*fcn*, *cfg*, *sections=[]*)

Get relevant function arguments given a configuration file

`config.`**`parse_config`**(*sections=[]*)

Wrapper for parsing config file for specific function call

`config.`**`read_config`**(*cfgfile, defaults={'channels': {'enabled': True, 'methods': ['gabor', 'gaussian', 'sobel'], 'methods_params': {'frequencies': [0.05, 0.15, 0.25], 'sigmas': [1, 8, 15], 'thetas': [0.0, 0.785, 1.571, 2.356]}}, 'segmentation': {'remove_disjoint': True, 'extract_contours': False, 'enabled': True, 'method': 'slic', 'method_params': {}}, 'relative_location': {'sigma': 2, 'enabled': False, 'n': 100}, 'features': {'feature_blocks': 'all', 'enabled': True, 'blocks_params': {}}, 'score': {}, 'partition': {'n_partitions': 5, 'force_split': False, 'enabled': True, 'frac_test': 0.25, 'frac_validation': 0.0}, 'training': {'partitions': 'all'}, 'regularization': {'C': [0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0], 'partition': 0}, 'general': {'model_type': 'LR', 'colorspace': 'rgb', 'class_aggregation': '', 'model_dataset': ''}}*)

Read configuration file and update default settings

`config.`**`write_config`**(*cfgfile, defaults={'channels': {'enabled': True, 'methods': ['gabor', 'gaussian', 'sobel'], 'methods_params': {'frequencies': [0.05, 0.15, 0.25], 'sigmas': [1, 8, 15], 'thetas': [0.0, 0.785, 1.571, 2.356]}}, 'segmentation': {'remove_disjoint': True, 'extract_contours': False, 'enabled': True, 'method': 'slic', 'method_params': {}}, 'relative_location': {'sigma': 2, 'enabled': False, 'n': 100}, 'features': {'feature_blocks': 'all', 'enabled': True, 'blocks_params': {}}, 'score': {}, 'partition': {'n_partitions': 5, 'force_split': False, 'enabled': True, 'frac_test': 0.25, 'frac_validation': 0.0}, 'training': {'partitions': 'all'}, 'regularization': {'C': [0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0], 'partition': 0}, 'general': {'model_type': 'LR', 'colorspace': 'rgb', 'class_aggregation': '', 'model_dataset': ''}}*)

Write configuration file

Example configuration

```
[channels]
enabled = True
methods = ["gabor", "gaussian", "sobel"]
methods_params = {"frequencies": [0.05, 0.15, 0.25], "sigmas": [1, 8, 15], "thetas": [0.0, 0.785, 1.5

[segmentation]
remove_disjoint = True
extract_contours = False
enabled = True
method = slic
method_params = {}

[relative_location]
sigma = 2
enabled = False
n = 100

[features]
feature_blocks = all
enabled = True
blocks_params = {}

[score]

[train]
partitions = all

[partition]
n_partitions = 5
force_split = False
enabled = True
frac_test = 0.25
frac_validation = 0.0

[regularization]
c = [0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0]
partition = 0

[general]
model_type = LR
colorspace = rgb
class_aggregation =
model_dataset =
```

CHAPTER 5

Acknowledgements

The Flamingo toolbox is developed at Delft University of Technology with support from the ERC-Advanced Grant 291206 – Nearshore Monitoring and Modeling (NEMO), STW Grant 12686 – Nature-driven Nourishments of Coastal Systems (NatureCoast), S1: Coastal safety and Deltares.

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*

Python Module Index

Index