

Colour - Checker Detection Documentation

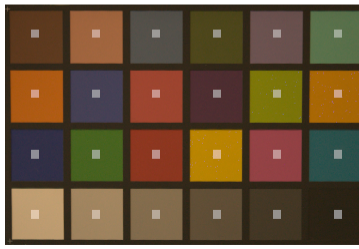
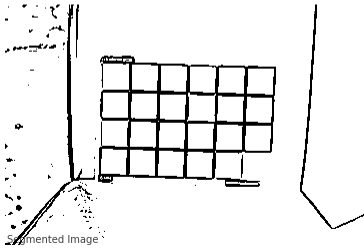
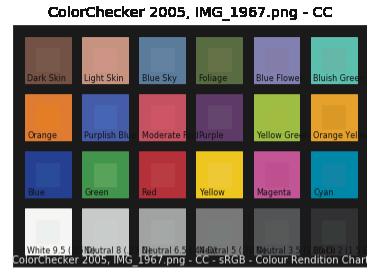
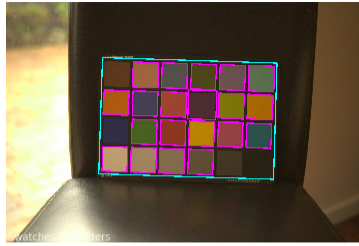
Release 0.1.0

Colour Developers

Apr 26, 2019

Contents

1	Features	3
2	Installation	5
2.1	Primary Dependencies	5
2.2	Pypi	5
3	Usage	7
3.1	API	7
3.1.1	Colour - Checker Detection Manual	7
3.1.1.1	Reference	7
3.1.1.2	Bibliography	11
3.2	Examples	11
4	Contributing	13
5	Bibliography	15
6	About	17
	Bibliography	19



A [Python](#) package implementing various colour checker detection algorithms and related utilities. It is open source and freely available under the [New BSD License](#) terms.

CHAPTER 1

Features

The following colour checker detection algorithms are implemented:

- Segmentation

Because of their size, the resources dependencies needed to run the various examples and unit tests are not provided within the Pypi package. They are separately available as [Git Submodules](#) when cloning the repository.

2.1 Primary Dependencies

Colour - Checker Detection requires various dependencies in order to run:

- Python 2.7 or Python 3.5
- Colour Science
- NumPy
- OpenImageIO

2.2 Pypi

Once the dependencies satisfied, **Colour - Checker Detection** can be installed from the [Python Package Index](#) by issuing this command in a shell:

```
pip install colour-checker-detection
```

The tests suite dependencies are installed as follows:

```
pip install 'colour-checker-detection[tests]'
```

The documentation building dependencies are installed as follows:

```
pip install 'colour-checker-detection[docs]'
```

The overall development dependencies are installed as follows:

```
pip install 'colour-checker-detection[development]'
```

3.1 API

The main reference for Colour - Demosaicing is the manual:

3.1.1 Colour - Checker Detection Manual

3.1.1.1 Reference

Colour - Checker Detection

Colour Checker Detection

- *Segmentation*

Segmentation

colour_checker_detection

<code>colour_checkers_coordinates_segmentation(image)</code>	Detects the colour checkers coordinates in given image <i>image</i> using segmentation.
<code>extract_colour_checkers_segmentation(image)</code>	Extracts the colour checkers sub-images in given image using segmentation.
<code>detect_colour_checkers_segmentation(image[, ...])</code>	Detects the colour checkers swatches in given image using segmentation.

colour_checker_detection.colour_checkers_coordinates_segmentation

colour_checker_detection.colour_checkers_coordinates_segmentation(*image*, *additional_data=False*)

Detects the colour checkers coordinates in given image *image* using segmentation.

This is the core detection definition. The process is as follows:

- Input image *image* is converted to a grayscale image *image_g*.
- Image *image_g* is denoised.
- Image *image_g* is thresholded/segmented to image *image_s*.
- Image *image_s* is eroded and dilated to cleanup remaining noise.
- Contours are detected on image *image_s*.
- Contours are filtered to only keep squares/swatches above and below defined surface area.
- Squares/swatches are clustered to isolate region-of-interest that are potentially colour checkers: Contours are scaled by a third so that colour checkers swatches are expected to be joined, creating a large rectangular cluster. Rectangles are fitted to the clusters.
- Clusters with an aspect ratio different to the expected one are rejected, a side-effect is that the complementary pane of the *X-Rite ColorChecker Passport* is omitted.
- Clusters with a number of swatches close to SWATCHES are kept.

Parameters

- **image** (array_like) – Image to detect the colour checkers in.
- **additional_data** (bool, optional) – Whether to output additional data.

Returns List of colour checkers coordinates or ColourCheckersDetectionData class instance with additional data.

Return type list or ColourCheckersDetectionData

Notes

- Multiple colour checkers can be detected if presented in image.

Examples

```
>>> import os
>>> from colour import read_image
>>> from colour_checker_detection import TESTS_RESOURCES_DIRECTORY
>>> path = os.path.join(TESTS_RESOURCES_DIRECTORY,
...                     'colour_checker_detection', 'detection',
...                     'IMG_1967.png')
>>> image = read_image(path)
>>> colour_checkers_coordinates_segmentation(image)
[array([[1065, 707],
       [ 369, 688],
       [ 382, 226],
       [1078, 246]])]
```

`colour_checker_detection.extract_colour_checkers_segmentation``colour_checker_detection.extract_colour_checkers_segmentation(image)`

Extracts the colour checkers sub-images in given image using segmentation.

Parameters `image` (array_like) – Image to extract the colours checkers sub-images from.**Returns** List of colour checkers sub-images.**Return type** `list`**Examples**

```

>>> import os
>>> from colour import read_image
>>> from colour_checker_detection import TESTS_RESOURCES_DIRECTORY
>>> path = os.path.join(TESTS_RESOURCES_DIRECTORY,
...                     'colour_checker_detection', 'detection',
...                     'IMG_1967.png')
>>> image = read_image(path)
>>> extract_colour_checkers_segmentation(image)
... # doctest: +SKIP
[array([[ 86, 104, 113],
       [ 89, 102, 118],
       [ 88, 101, 117],
       ...,
       [ 79, 101, 114],
       [ 76, 101, 114],
       [ 79, 98, 110]],
<BLANKLINE>
       [[ 84, 104, 112],
       [ 85, 102, 115],
       [ 84, 101, 115],
       ...,
       [ 80, 101, 110],
       [ 79, 101, 112],
       [ 78, 98, 112]],
<BLANKLINE>
       [[ 84, 102, 112],
       [ 82, 102, 112],
       [ 82, 101, 113],
       ...,
       [ 81, 100, 109],
       [ 80, 100, 110],
       [ 79, 100, 113]],
<BLANKLINE>
       ...,
       [[ 89, 105, 117],
       [ 90, 106, 120],
       [ 86, 106, 117],
       ...,
       [ 84, 100, 109],
       [ 83, 100, 111],
       [ 80, 100, 114]],
<BLANKLINE>
       [[ 89, 106, 116],
       [ 91, 107, 121],

```

(continues on next page)

(continued from previous page)

```

    [ 89, 106, 119],
    ...,
    [ 81, 99, 113],
    [ 79, 100, 115],
    [ 75, 100, 114]],
<BLANKLINE>
[[ [ 84, 108, 117],
   [ 89, 108, 117],
   [ 91, 107, 117],
   ...,
   [ 79, 98, 117],
   [ 77, 100, 117],
   [ 73, 101, 116]]], dtype=uint8)]

```

colour_checker_detection.detect_colour_checkers_segmentation

colour_checker_detection.detect_colour_checkers_segmentation(*image*, *samples*=16, *additional_data*=False)

Detects the colour checkers swatches in given image using segmentation.

Parameters

- **image** (array_like) – Image to detect the colour checkers swatches in.
- **samples** (int) – Samples count to use to compute the swatches colours. The effective samples count is $samples^2$.
- **additional_data** (bool, optional) – Whether to output additional data.

Returns List of colour checkers swatches or ColourCheckerSwatchesData class instances.

Return type list

Examples

```

>>> import os
>>> from colour import read_image
>>> from colour_checker_detection import TESTS_RESOURCES_DIRECTORY
>>> path = os.path.join(TESTS_RESOURCES_DIRECTORY,
...                     'colour_checker_detection', 'detection',
...                     'IMG_1967.png')
>>> image = read_image(path)
>>> detect_colour_checkers_segmentation(image) # doctest: +ELLIPSIS
[array([[ 0.3594894...,  0.2225419...,  0.1176996...],
        [ 0.6250058...,  0.3931947...,  0.2417636...],
        [ 0.3304194...,  0.3142103...,  0.2874383...],
        [ 0.3034269...,  0.2721812...,  0.1053537...],
        [ 0.4153488...,  0.3183605...,  0.3067842...],
        [ 0.3458465...,  0.4393400...,  0.2912665...],
        [ 0.6782215...,  0.3519573...,  0.0752686...],
        [ 0.2715231...,  0.2515535...,  0.3295411...],
        [ 0.6171124...,  0.2687208...,  0.1852935...],
        [ 0.3049796...,  0.1792275...,  0.1908085...],
        [ 0.4844366...,  0.4576518...,  0.0392559...],
        [ 0.6494152...,  0.3991223...,  0.0329260...],

```

(continues on next page)

(continued from previous page)

```
[ 0.1922949..., 0.1842026..., 0.2731065...],  
[ 0.2780555..., 0.3836590..., 0.1233134...],  
[ 0.5515815..., 0.2126631..., 0.1250530...],  
[ 0.7178619..., 0.5132913..., 0.0804213...],  
[ 0.5753956..., 0.2563947..., 0.2672106...],  
[ 0.1799058..., 0.3160584..., 0.2945296...],  
[ 0.7402078..., 0.6088296..., 0.4374975...],  
[ 0.6272391..., 0.5156084..., 0.3713541...],  
[ 0.5120363..., 0.4196305..., 0.2976295...],  
[ 0.3690167..., 0.3019190..., 0.2083050...],  
[ 0.2624792..., 0.2143349..., 0.1428991...],  
[ 0.1625438..., 0.1333312..., 0.0807412...]]]
```

Indices and tables

- [genindex](#)
- [search](#)

3.1.1.2 Bibliography

3.2 Examples

Various usage examples are available from the [examples directory](#).

CHAPTER 4

Contributing

If you would like to contribute to [Colour - Checker Detection](#), please refer to the following [Contributing guide for Colour](#).

CHAPTER 5

Bibliography

The bibliography is available in the repository in [BibTeX](#) format.

CHAPTER 6

About

Colour - Checker Detection by Colour Developers

Copyright © 2018-2019 – Colour Developers – colour-science@googlegroups.com

This software is released under terms of New BSD License: <http://opensource.org/licenses/BSD-3-Clause>

<http://github.com/colour-science/colour-checker-detection>

Bibliography

- [Abe11] Felix Abecassis. OpenCV - Rotation (Deskewing). 2011. URL: <http://felix.abecassis.me/2011/10/opencv-rotation-deskewing/>.

C

`colour_checkers_coordinates_segmentation()` (*in module `colour_checker_detection`*), 8

D

`detect_colour_checkers_segmentation()` (*in module `colour_checker_detection`*), 10

E

`extract_colour_checkers_segmentation()` (*in module `colour_checker_detection`*), 9