

**COLOUR - CHECKER DETECTION**

# **Colour - Checker Detection Documentation**

*Release 0.1.4*

**Colour Developers**

**Dec 04, 2022**



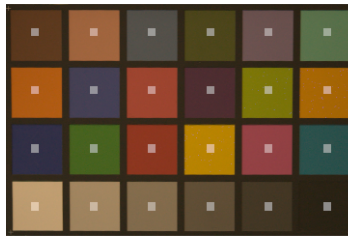
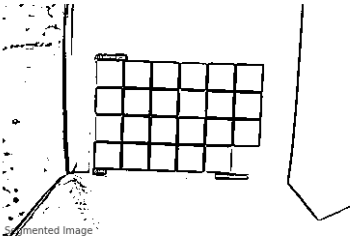
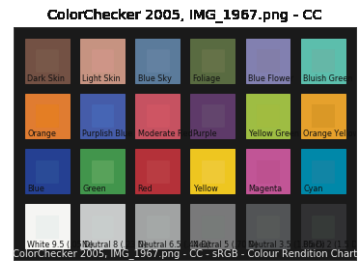
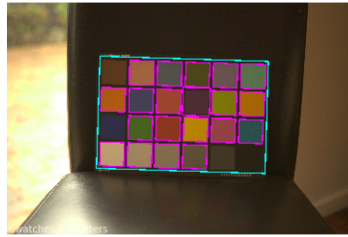
## CONTENTS

|          |            |                             |           |
|----------|------------|-----------------------------|-----------|
| <b>1</b> | <b>1.1</b> | <b>Features</b>             | <b>3</b>  |
|          | 1.1        | 1.1.1 Examples . . . . .    | 3         |
| <b>2</b> | <b>1.2</b> | <b>User Guide</b>           | <b>5</b>  |
|          | 2.1        | User Guide . . . . .        | 5         |
| <b>3</b> | <b>1.3</b> | <b>API Reference</b>        | <b>7</b>  |
|          | 3.1        | API Reference . . . . .     | 7         |
| <b>4</b> | <b>1.4</b> | <b>Code of Conduct</b>      | <b>15</b> |
| <b>5</b> | <b>1.5</b> | <b>Contact &amp; Social</b> | <b>17</b> |
| <b>6</b> | <b>1.6</b> | <b>About</b>                | <b>19</b> |
|          |            | <b>Bibliography</b>         | <b>21</b> |
|          |            | <b>Index</b>                | <b>23</b> |



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.





## 1.1 FEATURES

The following colour checker detection algorithms are implemented:

- Segmentation

### 1.1 1.1.1 Examples

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





## 1.2 USER GUIDE

### 2.1 User Guide

The user guide provides an overview of **Colour - Checker Detection** and explains important concepts and features, details can be found in the [API Reference](#).

#### 2.1.1 Installation Guide

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](#).

##### Primary Dependencies

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

- `python >= 3.8, < 4`
- `colour-science >= 4`
- `imageio >= 2, < 3`
- `numpy >= 1.19, < 2`
- `opencv-python >= 4, < 5`
- `scipy >= 1.5, < 2`

##### Pypi

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

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

The tests suite dependencies are installed as follows:

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

The documentation building dependencies are installed as follows:

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

The overall development dependencies are installed as follows:

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

## 2.1.2 Bibliography

## 1.3 API REFERENCE

### 3.1 API Reference

#### 3.1.1 Colour - Checker Detection

##### Colour Checker Detection

##### Segmentation

colour\_checker\_detection

|  |   |
|--|---|
| <code>SETTINGS_SEGMENTATION_COLORCHECKER_CLASSIC</code>        | Settings for the segmentation of the <i>X-Rite ColorChecker Classic</i> and <i>X-Rite ColorChecker Passport</i> . |
| <code>SETTINGS_SEGMENTATION_COLORCHECKER_SG</code>             | Settings for the segmentation of the <i>X-Rite ColorChecker SG*</i> .   |
| <code>colour_checkers_coordinates_segmentation(image)</code>   | Detect the colour checkers coordinates in given image <i>image</i> using segmentation.                            |
| <code>extract_colour_checkers_segmentation(image, ...)</code>  | Extract the colour checkers sub-images in given image using segmentation.   |
| <code>detect_colour_checkers_segmentation(image[, ...])</code> | Detect the colour checkers swatches in given image using segmentation.  |

##### colour\_checker\_detection.SETTINGS\_SEGMENTATION\_COLORCHECKER\_CLASSIC

```
colour_checker_detection.SETTINGS_SEGMENTATION_COLORCHECKER_CLASSIC =  
{  
    'adaptive_threshold_kwargs': {'C': 3, 'adaptiveMethod': 0, 'blockSize': 21, 'maxValue':  
255, 'thresholdType': 0}, 'aspect_ratio': 1.5, 'aspect_ratio_maximum': 1.6500000000000001,  
'aspect_ratio_minimum': 1.35, 'cluster_contour_scale': 0.975,  
'fast_non_local_means_denoising_kwargs': {'h': 10, 'searchWindowSize': 21,  
'templateWindowSize': 7}, 'interpolation_method': 2, 'swatch_contour_scale':  
1.3333333333333333, 'swatch_minimum_area_factor': 200, 'swatches': 24,  
'swatches_achromatic_slice': slice(19, 23, 1), 'swatches_chromatic_slice': slice(1, 5, 1),  
'swatches_count_maximum': 30, 'swatches_count_minimum': 18, 'swatches_horizontal': 6,  
'swatches_vertical': 4, 'working_width': 1440}
```

Settings for the segmentation of the *X-Rite ColorChecker Classic* and *X-Rite ColorChecker Passport*.

### colour\_checker\_detection.SETTINGS\_SEGMENTATION\_COLORCHECKER\_SG

```
colour_checker_detection.SETTINGS_SEGMENTATION_COLORCHECKER_SG =  
{'adaptive_threshold_kwargs': {'C': 3, 'adaptiveMethod': 0, 'blockSize': 21, 'maxValue':  
255, 'thresholdType': 0}, 'aspect_ratio': 1.4, 'aspect_ratio_maximum': 1.54,  
'aspect_ratio_minimum': 1.26, 'cluster_contour_scale': 1,  
'fast_non_local_means_denoising_kwargs': {'h': 10, 'searchWindowSize': 21,  
'templateWindowSize': 7}, 'interpolation_method': 2, 'swatch_contour_scale':  
1.3333333333333333, 'swatch_minimum_area_factor': 200, 'swatches': 140,  
'swatches_achromatic_slice': slice(115, 120, 1), 'swatches_chromatic_slice': slice(48, 53,  
1), 'swatches_count_maximum': 210, 'swatches_count_minimum': 70, 'swatches_horizontal':  
14, 'swatches_vertical': 10, 'working_width': 1440}
```

Settings for the segmentation of the *X-Rite ColorChecker SG*.\*

### colour\_checker\_detection.colour\_checkers\_coordinates\_segmentation

```
colour_checker_detection.colour_checkers_coordinates_segmentation(image: ArrayLike,  
                                                                    additional_data: bool =  
                                                                    False, **kwargs: Any) →  
                                                                    Union[DataColourCheckersCoordinatesSegm  
                                                                    Tuple[ndarray, ...]]
```

Detect 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<sub>g</sub>*.
- Image *image<sub>g</sub>* is denoised.
- Image *image<sub>g</sub>* is thresholded/segmented to image *image<sub>s</sub>*.
- Image *image<sub>s</sub>* is eroded and dilated to cleanup remaining noise.
- Contours are detected on image *image<sub>s</sub>*.
- 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 checker 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 the expected one are kept.

#### Parameters

- **image** (ArrayLike) – Image to detect the colour checkers in.
- **additional\_data** (bool) – Whether to output additional data.
- **aspect\_ratio** – Colour checker aspect ratio, e.g. 1.5.
- **aspect\_ratio\_minimum** – Minimum colour checker aspect ratio for detection: projective geometry might reduce the colour checker aspect ratio.
- **aspect\_ratio\_maximum** – Maximum colour checker aspect ratio for detection: projective geometry might increase the colour checker aspect ratio.
- **swatches** – Colour checker swatches total count.
- **swatches\_horizontal** – Colour checker swatches horizontal columns count.
- **swatches\_vertical** – Colour checker swatches vertical row count.

- **swatches\_count\_minimum** – Minimum swatches count to be considered for the detection.
- **swatches\_count\_maximum** – Maximum swatches count to be considered for the detection.
- **swatches\_chromatic\_slice** – A *slice* instance defining chromatic swatches used to detect if the colour checker is upside down.
- **swatches\_achromatic\_slice** – A *slice* instance defining achromatic swatches used to detect if the colour checker is upside down.
- **swatch\_minimum\_area\_factor** – Swatch minimum area factor  $f$  with the minimum area  $m_a$  expressed as follows:  $m_a = image_w * image_h / s_c / f$  where  $image_w$ ,  $image_h$  and  $s_c$  are respectively the image width, height and the swatches count.
- **swatch\_contour\_scale** – As the image is filtered, the swatches area will tend to shrink, the generated contours can thus be scaled.
- **cluster\_contour\_scale** – As the swatches are clustered, it might be necessary to adjust the cluster scale so that the masks are centred better on the swatches.
- **working\_width** – Size the input image is resized to for detection.
- **fast\_non\_local\_means\_denoising\_kwargs** – Keyword arguments for `cv2.fastNlMeansDenoising()` definition.
- **adaptive\_threshold\_kwargs** – Keyword arguments for `cv2.adaptiveThreshold()` definition.
- **interpolation\_method** – Interpolation method used when resizing the images, `cv2.INTER_CUBIC` and `cv2.INTER_LINEAR` methods are recommended.
- **kwargs** (Any) –

#### Returns

Tuple of colour checkers coordinates or `DataColourCheckersCoordinatesSegmentation` class instance with additional data.

#### Return type

`colour_checker_detection.detection.segmentation`.  
`DataColourCheckersCoordinatesSegmentation` or `tuple`

#### Notes

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

#### Examples

```
>>> import os
>>> from colour import read_image
>>> from colour_checker_detection import ROOT_RESOURCES_TESTS
>>> path = os.path.join(
...     ROOT_RESOURCES_TESTS,
...     "colour_checker_detection",
...     "detection",
...     "IMG_1967.png",
... )
>>> image = read_image(path)
>>> colour_checkers_coordinates_segmentation(image)
(array([[ 369,  688],
        [ 382,  226],
```

(continues on next page)

(continued from previous page)

```
[1078, 246],  
[1065, 707]]...)
```

### colour\_checker\_detection.extract\_colour\_checkers\_segmentation

colour\_checker\_detection.extract\_colour\_checkers\_segmentation(*image*: ArrayLike, *\*\*kwargs*: Any) → Tuple[ndarray, ...]

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

#### Parameters

- **image** (ArrayLike) – Image to extract the colours checkers sub-images from.
- **aspect\_ratio** – Colour checker aspect ratio, e.g. 1.5.
- **aspect\_ratio\_minimum** – Minimum colour checker aspect ratio for detection: projective geometry might reduce the colour checker aspect ratio.
- **aspect\_ratio\_maximum** – Maximum colour checker aspect ratio for detection: projective geometry might increase the colour checker aspect ratio.
- **swatches** – Colour checker swatches total count.
- **swatches\_horizontal** – Colour checker swatches horizontal columns count.
- **swatches\_vertical** – Colour checker swatches vertical row count.
- **swatches\_count\_minimum** – Minimum swatches count to be considered for the detection.
- **swatches\_count\_maximum** – Maximum swatches count to be considered for the detection.
- **swatches\_chromatic\_slice** – A *slice* instance defining chromatic swatches used to detect if the colour checker is upside down.
- **swatches\_achromatic\_slice** – A *slice* instance defining achromatic swatches used to detect if the colour checker is upside down.
- **swatch\_minimum\_area\_factor** – Swatch minimum area factor  $f$  with the minimum area  $m_a$  expressed as follows:  $m_a = image_w * image_h / s_c / f$  where  $image_w$ ,  $image_h$  and  $s_c$  are respectively the image width, height and the swatches count.
- **swatch\_contour\_scale** – As the image is filtered, the swatches area will tend to shrink, the generated contours can thus be scaled.
- **cluster\_contour\_scale** – As the swatches are clustered, it might be necessary to adjust the cluster scale so that the masks are centred better on the swatches.
- **working\_width** – Size the input image is resized to for detection.
- **fast\_non\_local\_means\_denoising\_kwargs** – Keyword arguments for cv2.fastNlMeansDenoising() definition.
- **adaptive\_threshold\_kwargs** – Keyword arguments for cv2.adaptiveThreshold() definition.
- **interpolation\_method** – Interpolation method used when resizing the images, cv2.INTER\_CUBIC and cv2.INTER\_LINEAR methods are recommended.
- **kwargs** (Any) –

#### Returns

Tuple of colour checkers sub-images.

**Return type**

tuple

**Examples**

```

>>> import os
>>> from colour import read_image
>>> from colour_checker_detection import ROOT_RESOURCES_TESTS
>>> path = os.path.join(
...     ROOT_RESOURCES_TESTS,
...     "colour_checker_detection",
...     "detection",
...     "IMG_1967.png",
... )
>>> image = read_image(path)
>>> extract_colour_checkers_segmentation(image)
...
(array([[ 0.17908671,  0.14010708,  0.09243158],
        [ 0.17805016,  0.13058874,  0.09513047],
        [ 0.17175764,  0.13128328,  0.08811688],
        ...,
        [ 0.15934898,  0.13436384,  0.07479276],
        [ 0.17178158,  0.13138185,  0.07703256],
        [ 0.15082785,  0.11866678,  0.07680314]],

        [[ 0.16597673,  0.13563241,  0.08780421],
        [ 0.16490564,  0.13110894,  0.08601525],
        [ 0.16939694,  0.12963502,  0.08783565],
        ...,
        [ 0.14708202,  0.12856133,  0.0814603 ],
        [ 0.16883563,  0.12862256,  0.08452422],
        [ 0.16781917,  0.12363558,  0.07361614]],

        [[ 0.16326806,  0.13720085,  0.08925959],
        [ 0.16014062,  0.13585283,  0.08104862],
        [ 0.16657823,  0.12889633,  0.08870038],
        ...,
        [ 0.14619341,  0.13086307,  0.07367594],
        [ 0.16302426,  0.13062705,  0.07938427],
        [ 0.16618022,  0.1266259 ,  0.07200021]],

        ...,
        [[ 0.1928642 ,  0.14578913,  0.11224515],
        [ 0.18931177,  0.14416392,  0.10288388],
        [ 0.17707473,  0.1436448 ,  0.09188452],
        ...,
        [ 0.16879168,  0.12867133,  0.09001681],
        [ 0.1699731 ,  0.1287041 ,  0.07616285],
        [ 0.17137891,  0.129711 ,  0.07517841]],

        [[ 0.19514292,  0.1532704 ,  0.10375113],
        [ 0.18217109,  0.14982903,  0.10452617],
        [ 0.18830594,  0.1469499 ,  0.10896181],
        ...,
        [ 0.18234864,  0.12642328,  0.08047272],
        [ 0.17617388,  0.13000189,  0.06874527],

```

(continues on next page)

(continued from previous page)

```
[ 0.17108543, 0.13264084, 0.06309374]],
[[ 0.16243187, 0.14983535, 0.08954653],
 [ 0.155507 , 0.14899652, 0.10273992],
 [ 0.17993385, 0.1498394 , 0.1099571 ],
 ...,
 [ 0.18079454, 0.1253967 , 0.07739887],
 [ 0.17239226, 0.13181566, 0.07806754],
 [ 0.17422497, 0.13277327, 0.07513551]]], dtype=float32),)
```

### colour\_checker\_detection.detect\_colour\_checkers\_segmentation

`colour_checker_detection.detect_colour_checkers_segmentation(image: ArrayLike, samples: int = 16, additional_data: bool = False, **kwargs: Any) → Union[Tuple[DataDetectColourCheckersSegmentation, Tuple[ndarray, ...]]`

Detect 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.
- **aspect\_ratio** – Colour checker aspect ratio, e.g. 1.5.
- **aspect\_ratio\_minimum** – Minimum colour checker aspect ratio for detection: projective geometry might reduce the colour checker aspect ratio.
- **aspect\_ratio\_maximum** – Maximum colour checker aspect ratio for detection: projective geometry might increase the colour checker aspect ratio.
- **swatches** – Colour checker swatches total count.
- **swatches\_horizontal** – Colour checker swatches horizontal columns count.
- **swatches\_vertical** – Colour checker swatches vertical row count.
- **swatches\_count\_minimum** – Minimum swatches count to be considered for the detection.
- **swatches\_count\_maximum** – Maximum swatches count to be considered for the detection.
- **swatches\_chromatic\_slice** – A *slice* instance defining chromatic swatches used to detect if the colour checker is upside down.
- **swatches\_achromatic\_slice** – A *slice* instance defining achromatic swatches used to detect if the colour checker is upside down.
- **swatch\_minimum\_area\_factor** – Swatch minimum area factor  $f$  with the minimum area  $m_a$  expressed as follows:  $m_a = image_w * image_h / s_c / f$  where  $image_w$ ,  $image_h$  and  $s_c$  are respectively the image width, height and the swatches count.
- **swatch\_contour\_scale** – As the image is filtered, the swatches area will tend to shrink, the generated contours can thus be scaled.
- **cluster\_contour\_scale** – As the swatches are clustered, it might be necessary to adjust the cluster scale so that the masks are centred better on the swatches.



- **working\_width** – Size the input image is resized to for detection.
- **fast\_non\_local\_means\_denoising\_kwargs** – Keyword arguments for `cv2.fastNlMeansDenoising()` definition.
- **adaptive\_threshold\_kwargs** – Keyword arguments for `cv2.adaptiveThreshold()` definition.
- **interpolation\_method** – Interpolation method used when resizing the images, `cv2.INTER_CUBIC` and `cv2.INTER_LINEAR` methods are recommended.
- **kwargs** (*Any*) –

**Returns**

Tuple of `DataDetectColourCheckersSegmentation` class instances or colour checkers swatches.

**Return type**

class`tuple`

**Examples**

```
>>> import os
>>> from colour import read_image
>>> from colour_checker_detection import ROOT_RESOURCES_TESTS
>>> path = os.path.join(
...     ROOT_RESOURCES_TESTS,
...     "colour_checker_detection",
...     "detection",
...     "IMG_1967.png",
... )
>>> image = read_image(path)
>>> detect_colour_checkers_segmentation(image)
(array([[ 0.361626...,  0.2241066...,  0.1187837...],
       [ 0.6280594...,  0.3950883...,  0.2434766...],
       [ 0.3326232...,  0.3156182...,  0.2891038...],
       [ 0.3048414...,  0.2738973...,  0.1069985...],
       [ 0.4174869...,  0.3199669...,  0.3081552...],
       [ 0.347873 ...,  0.4413193...,  0.2931614...],
       [ 0.6816301...,  0.3539050...,  0.0753397...],
       [ 0.2731050...,  0.2528467...,  0.3312920...],
       [ 0.6192335...,  0.2703833...,  0.1866387...],
       [ 0.3068567...,  0.1803366...,  0.1919807...],
       [ 0.4866354...,  0.4594004...,  0.0374186...],
       [ 0.6518523...,  0.4010608...,  0.0171886...],
       [ 0.1941571...,  0.1855801...,  0.2750632...],
       [ 0.2799946...,  0.3854609...,  0.1241038...],
       [ 0.5537481...,  0.2139004...,  0.1267332...],
       [ 0.7208045...,  0.5152904...,  0.0061946...],
       [ 0.5778360...,  0.2578533...,  0.2687992...],
       [ 0.1809450...,  0.3174742...,  0.2959902...],
       [ 0.7427522...,  0.6107554...,  0.4398439...],
       [ 0.6296108...,  0.5177606...,  0.3728032...],
       [ 0.5139589...,  0.4216307...,  0.2992694...],
       [ 0.3704401...,  0.3033927...,  0.2093089...],
       [ 0.2641854...,  0.2154007...,  0.1441267...],
       [ 0.1650098...,  0.1345239...,  0.0817437...]], dtype=float32),)
```

### 3.1.2 Indices and tables

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

## 1.4 CODE OF CONDUCT

The *Code of Conduct*, adapted from the [Contributor Covenant 1.4](#), is available on the [Code of Conduct](#) page.



## 1.5 CONTACT & SOCIAL

The *Colour Developers* can be reached via different means:

- [Email](#)
- [Facebook](#)
- [Github Discussions](#)
- [Gitter](#)
- [Twitter](#)



## 1.6 ABOUT

**Colour - Checker Detection** by Colour Developers

Copyright 2018 Colour Developers – [colour-developers@colour-science.org](mailto:colour-developers@colour-science.org)

This software is released under terms of New BSD License:

<https://opensource.org/licenses/BSD-3-Clause>

<https://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/> (visited on 2018-10-27).



## INDEX

### C

`colour_checkers_coordinates_segmentation()`  
(in module *colour\_checker\_detection*), 8

### D

`detect_colour_checkers_segmentation()` (in  
module *colour\_checker\_detection*), 12

### E

`extract_colour_checkers_segmentation()` (in  
module *colour\_checker\_detection*), 10

### S

`SETTINGS_SEGMENTATION_COLORCHECKER_CLASSIC`  
(in module *colour\_checker\_detection*), 7

`SETTINGS_SEGMENTATION_COLORCHECKER_SG` (in  
module *colour\_checker\_detection*), 8