ColourPaletteExtractor

Release 0.5.4

Tim Churchfield

Aug 01, 2021

Table of Contents

Chapter 1

colourpaletteextractor

1.1 colourpaletteextractor package

1.1.1 Subpackages

colourpaletteextractor.controller package

Submodules

colourpaletteextractor.controller.controller module

class `colourpaletteextractor.controller.controller.`**`ColourPaletteExtractorController`** (*model: colourpaletteextractor.model.model.ColourPaletteExtractorModel*, *view: colourpaletteextractor.view.-mainview.MainView*)

Bases: `PySide2.QtCore.QRunnable`

ColourPaletteExtractor Controller.

Used to connect the ColourPaletteExtractor GUI signals with the appropriate slot to be able to manipulate the associated model.

Parameters
- **`model`** (*`ColourPaletteExtractorModel`*) – The main model of ColourPaletteExtractor.
- **`view`** (*`MainView`*) – The main window of ColourPaletteExtractor.

`current_tab_changed` (*i: int*)

Update the current tab index and update the view with the tab's properties.

In most cases, i >= 0, however a value of i = -2 or -3 is also valid for performing a 'dummy' tab change to update the current view shown to the user. A value of -1 will lead to the creation of the default tab (the quick start guide).

Parameters **`i`** (*`int`*) – Index of the current tab.

Raises **`ValueError`** – If the value of i is less than -3.

colourpaletteextractor.controller.worker module

class `colourpaletteextractor.controller.worker.`**`Worker`** (*fn*, *function_type: str*, *tab: colourpaletteextractor.view.tabview.NewTab*, **args*, ***kwargs*)

Bases: `PySide2.QtCore.QRunnable`

Worker thread used to generate the colour palette or report for an image.

Inherits from QRunnable to handler worker thread setup, signals and wrap-up.

Adapted from: ref

Accessed: 01/08/21

Parameters
- **`fn`** – The function or method to be run as a new thread (generating an image's

colour palette or its colour palette report.

- **tab** (*NewTab*) – `tabview.NewTab` object associated with the image to be processed.
- **function_type** (*str*) – The action to be run. This can either be 'colour palette' or 'report'.
- ***args** – Arguments to pass to the callback function
- ***kwargs** – Keywords to pass to the callback function

Attributes:

Parameters **progress_callback** – The function callback to run on this worker thread. Supplied args and kwargs will be passed through to the runner.

Raises **ValueError** – If the provided function_type is invalid.

run ( )

Initialise the runner function with passed args, kwargs.

class `colourpaletteextractor.controller.worker.`**WorkerSignals**

Bases: `PySide2.QtCore.QObject`

Specify the signals available from a running `Worker` thread.

Adapted from: ref

Accessed: 01/08/21

Supported signals are:

error

Tuple (exc_type, value, traceback.format_exc()).

finished

Integer emitted upon finishing.

When generating a colour palette, the value is -2. When generating a report, the value is -3. This is used to reload the tab displaying the image with the correct settings and colour palette.

progress

NewTab object for which the GUI is to be updated for and the percentage complete for the current task.

The current task is either generating the colour palette for an image or generating the colour palette report for the image.

result

Object data returned from processing, anything - NOT IN USE.

staticMetaObject = <PySide2.QtCore.QMetaObject object>

Module contents

colourpaletteextractor.examples package

Submodules

colourpaletteextractor.examples.generatecolourpaletteexample module

Contains an example script demonstrating how to generate the colour palette of a sample image.

Module contents

colourpaletteextractor.model package

Subpackages

colourpaletteextractor.model.algorithms package

Submodules

colourpaletteextractor.model.algorithms.cielabcube module

class `colourpaletteextractor.model.algorithms.cielabcube.`**`CielabCube`** (*l_star_coord: int*, *a_star_coord: int*, *b_star_coord: int*)

Bases: `object`

A cube representing a fixed region in the CIELAB colour space.

The cube is used to hold pixels in an image that exist within the cube's region of the CIELAB colour space. The input parameters do not refer to the actual L*, a* and b* values, but depend on the *CUBE_SIZE* specified by the colour palette algorithm (in particular, any variant on the Nieves 2020 algorithm).

In the case of the `nieves2020.Nieves2020CentredCubes` algorithm, the coordinates refer to the centre of the cube. For the `nieves2020.Nieves2020OffsetCubes` algorithm, the coordinates refer to the corner of the cube closest to the origin.

Parameters
- **`l_star_coord`** (*`int`*) – Perceptual lightness cube coordinate
- **`a_star_coord`** (*`int`*) – Green-red cube coordinate
- **`b_star_coord`** (*`int`*) – Blue-yellow cube coordinate

`add_pixel_to_cube` (*pixel: numpy.array*, *c_star: numpy.float64*) → None

Assign a pixel to the cube.

Parameters
- **`pixel`** (*`np.array`*) – The pixel as a [L*,a*,b*] triplet.
- **`c_star`** (*`np.float64`*) – The C* (chroma, relative saturation) value for the pixel.

property `c_stars`: list

The C* (chroma, relative saturation) values for all of the pixels in the cube.

C^{*} = \sqrt{{a^{*}}^{2} + {b^{*}}^{2}}

Returns (list[np.float64]) – The list of C* values for all pixels in the cube.

`calculate_mean_colour` ( ) → None

Calculate the mean colour of the pixels in the cube.

Nothing is calculated if the number of pixels in the cube is equal to 0.

property `coordinates`: numpy.array

The coordinates of the cube ([L*, a*, b*]).

In the case of the `nieves2020.Nieves2020CentredCubes` algorithm, the coordinates refer to the centre of the cube. For the `nieves2020.Nieves2020OffsetCubes` algorithm, the coordinates refer to the corner of the cube closest to the origin.

Returns (np.array) – The cube's coordinates.

`get_c_star_percentile_value` (*percentile: float*) → Union[int, numpy.percentile]

Returns the C* value for the given percentile based on the pixels in the cube.

Parameters **`percentile`** (*`float`*) – The percentile to calculate the C* value for.

Returns **(Union[int, np.percentile])** – The C* value for the chosen percentile. If no pixels are found, the return value is 0.

`get_l_star_percentile_value` (*percentile: float*) → Union[int, numpy.percentile]

Returns the L* value for the given percentile based on the pixels in the cube.

Parameters **`percentile`** (*`float`*) – The percentile to calculate the L* value for.

Returns **(Union[int, np.percentile])** – The L* value for the chosen percentile. If no pixels are found, the return value is 0.

`increment_pixel_count_after_reassignment` ( ) → None
Increase the number of pixels with this cube's mean colour by one.

property `l_stars`: numpy.array
The L* values for all pixels in the cube.

Returns (np.array) – Array of L* values for all pixels in the cube.

property `mean_colour`: numpy.array
The mean colour of the pixels in the cube.

Returns (np.array) – The mean colour of the cube as a [L*,a*,b*] triplet.

property `pixel_count_after_reassignment`: int
The number of pixels in the recoloured image with this cube's mean colour.

Returns (int) – The number of pixels with the cube's mean colour.

property `pixels`: list
The list of pixels ([L*, a*, b*] triplets) in the cube.

Returns (list[np.array]) – The list of pixels in the cube.

property `relevant`: bool
The relevancy status of the cube.

Returns (bool) – True if the cube is a relevant cube. Otherwise False.

`colourpaletteextractor.model.algorithms.cielabcube.`**`get_relative_frequencies`** (*relevant_cubes: list*, *total_pixels: int*) → list
Calculate the relative frequency of each colour (relevant colour) in the recoloured image.

Parameters
- **`relevant_cubes`** (*`list[CielabCube]`*) – List of relevant `CielabCube` objects.
- **`total_pixels`** (*`int`*) – The total number of pixels in the image.

Returns **(list[float])** – The list of relative frequencies for each relevant cube.

colourpaletteextractor.model.algorithms.dummyalgorithm module

class `colourpaletteextractor.model.algorithms.dummyalgorithm.`**`TestAlgorithm`**
Bases: `colourpaletteextractor.model.algorithms.palettealgorithm.PaletteAlgorithm`

`generate_colour_palette` (*image*)
Generate the colour palette for the given image.
Analyses the given image to obtain its colour palette. Returns the recoloured image using only the colours found in the colour palette, the colour palette of the image and finally the relative frequencies of each of those colours in the recoloured image.

Parameters **`image`** (*`np.array`*) – A 3D array representing an image. It is assumed that the input image is

Returns
- **recoloured__image** (*np.array*) – The recoloured image using only the colours found in the colour palette
- **colour_palette** (*list*) – The list of colours (sRGB 8-bit values) in the colour palette
- **relative_frequencies** (*list*) – The relative frequencies of each colour in the colour palette in the recoloured image

Note It is assumed that the input image has been encoded in the sRGB colour space.

colourpaletteextractor.model.algorithms.grogan2018 module

class `colourpaletteextractor.model.algorithms.grogan2018.`**`Grogan2018`**

Bases: `colourpaletteextractor.model.algorithms.palettealgorithm.PaletteAlgorithm`

`generate_colour_palette` (*image*)

Generate the colour palette for the given image.

Analyses the given image to obtain its colour palette. Returns the recoloured image using only the colours found in the colour palette, the colour palette of the image and finally the relative frequencies of each of those colours in the recoloured image.

Parameters **`image`** (*`np.array`*) – A 3D array representing an image. It is assumed that the input image is

Returns

- **recoloured__image** (*np.array*) – The recoloured image using only the colours found in the colour palette
- **colour_palette** (*list*) – The list of colours (sRGB 8-bit values) in the colour palette
- **relative_frequencies** (*list*) – The relative frequencies of each colour in the colour palette in the recoloured image

Note It is assumed that the input image has been encoded in the sRGB colour space.

`name` = **'Grogan, Hudon, McCormack and Smolic (2018) [NOT IMPLEMENTED!]'**

`url` = **'https://v-sense.scss.tcd.ie/research/vfx-animation/automatic-palette-extraction-for-image-editing/'**

colourpaletteextractor.model.algorithms.nieves2020 module

class `colourpaletteextractor.model.algorithms.nieves2020.`**`Nieves2020`** (*name, url*)

Bases: `colourpaletteextractor.model.algorithms.palettealgorithm.PaletteAlgorithm`, `abc.ABC`

`COLOUR_CHANNELS` = **3**

`CUBE_SIZE` = **20**

`C_STAR_PERCENTILE` = **50**

`L_STAR_PERCENTILE_THRESHOLD` = **0.00375**

`MIN_L_STAR` = **80**

`SECONDARY_THRESHOLD` = **0.00375**

`THRESHOLD` = **0.03**

`generate_colour_palette` (*image*) → tuple

Generate the colour palette for the given image.

Analyses the given image to obtain its colour palette. Returns the recoloured image using only the colours found in the colour palette, the colour palette of the image and finally the relative frequencies of each of those colours in the recoloured image.

Parameters **`image`** (*`np.array`*) – A 3D array representing an image. It is assumed that the input image is

Returns

- **recoloured__image** (*np.array*) – The recoloured image using only the colours found in the colour palette
- **colour_palette** (*list*) – The list of colours (sRGB 8-bit values) in the colour palette
- **relative_frequencies** (*list*) – The relative frequencies of each colour in the colour palette in the recoloured image

---

Note It is assumed that the input image has been encoded in the sRGB colour space.

---

class
`colourpaletteextractor.model.algorithms.nieves2020.`**`Nieves2020CentredCubes`**

Bases: `colourpaletteextractor.model.algorithms.nieves2020.Nieves2020`

`name` **= 'Nieves, Gomez-Robledo, Chen and Romero (2020) - Cube centred on CIELAB origin'**

`url` **= 'https://doi.org/10.1364/AO.378659'**

class
`colourpaletteextractor.model.algorithms.nieves2020.`**`Nieves2020OffsetCubes`**

Bases: `colourpaletteextractor.model.algorithms.nieves2020.Nieves2020`

`name` **= 'Nieves, Gomez-Robledo, Chen and Romero (2020) - Cube corners at CIELAB origin'**

`url` **= 'https://doi.org/10.1364/AO.378659'**

`colourpaletteextractor.model.algorithms.nieves2020.`**`convert_lab_2_rgb`** (*image*)

Convert an image in the CIELAB colour space into the sRGB colour space.

`colourpaletteextractor.model.algorithms.nieves2020.`**`convert_rgb_2_lab`** (*image*)

Convert an sRBG image into the CIELAB colour space.

`colourpaletteextractor.model.algorithms.nieves2020.`**`get_c_stars`** (*lab*)

Return the matrix of C* (chroma) values for each pixel in the image.

colourpaletteextractor.model.algorithms.nieves2020cython module

colourpaletteextractor.model.algorithms.palettealgorithm module

class
`colourpaletteextractor.model.algorithms.palettealgorithm.`**`PaletteAlgorithm`** (*name: str*, *url: str*)

Bases: `abc.ABC`

Abstract class representing an algorithm used to obtain a colour palette from an image.

Parameters

- **`name`** (*`str`*) – Name of the algorithm
- **`url`** (*`str`*) – Link to a description of the algorithm

property **`continue_thread`**: **bool**

Get the execution status of the algorithm.

A value of *false* would indicate that the algorithm should return without generation a colour palette when it next checks its execution status.

Returns **bool** – The execution status of the algorithm abstract **`generate_colour_palette`** (*image: numpy.array*) → tuple

Generate the colour palette for the given image.

Analyses the given image to obtain its colour palette. Returns the recoloured image using only the colours found in the colour palette, the colour palette of the image and finally the relative frequencies of each of those colours in the recoloured image.

Parameters **`image`** (*`np.array`*) – A 3D array representing an image. It is assumed that the input image is

Returns

- **recoloured__image** (*np.array*) – The recoloured image using only the colours found in the colour palette
- **colour_palette** (*list*) – The list of colours (sRGB 8-bit values) in the colour palette
- **relative_frequencies** (*list*) – The relative frequencies of each colour in the colour palette in the recoloured image

Note It is assumed that the input image has been encoded in the sRGB colour space.

property `name`: str

Get the name of the algorithm.

Returns (str) – The name of the algorithm

`set_progress_callback` (*progress_callback: PySide2.QtCore.SignalInstance*, *tab: colourpaletteextractor.view.tabview.NewTab*, *image_data*) → None

Set the signal function called by the algorithm at regular intervals to update the GUI thread.

Parameters

- **`progress_callback`** (*`QtCore.SignalInstance`*) – Signal that when emitted, is used to update the GUI.
- **`tab`** (*`NewTab`*) – The tab associated with the image being analysed (see `generate_colour_palette()`.
- **`image_data`** (*`ImageData`*) – *ImageData* object that holds the image being analysed.

property `url`: str

Get the link to the description of the algorithm.

Returns (str) – The link to the description of the algorithm

`colourpaletteextractor.model.algorithms.palettealgorithm.`**`get_implemented_algorithms`** ( )

Recursively finds all subclasses of the `PaletteAlgorithm` class.

Like Python's __class__.__subclasses__(), but recursive. Returns a list containing all subclasses of `PaletteAlgorithm`.

Adapted from: ref

Accessed: 15/07/2021

Returns:

[object]: List of all non-abstract subclasses of `PaletteAlgorithm`

Module contents

Submodules

colourpaletteextractor.model.dummyimagescript module

colourpaletteextractor.model.generatereport module

class `colourpaletteextractor.model.generatereport.`**`ColourPaletteReport`** (*image_data: colourpaletteextractor.model.imagedata.ImageData*)

Bases: `fpdf.fpdf.FPDF`, `fpdf.html.HTMLMixin`

A modified FPDF object to fit the requirements for generating a PDF colour palette report.

Parameters **`image_data`** (*`ImageData`*) – The ImageData object holding the image's data (the original image, the recoloured image, and the colour palette).

`A4_HEIGHT` **= 297**

The height of an A4 sheet of paper (mm).

`A4_WIDTH` **= 210**

The width of an A4 sheet of paper (mm).

`IMAGE_START_POSITION` **= 30**

The standard left indentation when placing an image in the PDF report (mm).

`IMAGE_WIDTH` **= 150**

The standard width of images in the PDF report (mm).

`MARGIN` **= 10**

The size of the margins to be used in the PDF report (mm).

`MAX_IMAGE_HEIGHT` **= 257**

The standard maximum height of images in the report (mm).

`footer` ( ) → None

Set the footer used in the PDF report.

`header` ( ) → None

Set the header used in the PDF report.

class `colourpaletteextractor.model.generatereport.`**`ReportGenerator`** (*tab: colourpaletteextractor.view.tabview.NewTab*, *image_data: colourpaletteextractor.model.imagedata.ImageData*, *progress_callback: PySide2.QtCore.SignalInstance*)

Bases: `object`

Class used to create, populate a `ColourPaletteReport` object and save the resulting PDF to disk.

Parameters
- **`tab`** (*`NewTab`*) – The tab associated with the image to be analysed.
- **`image_data`** (*`ImageData`*) – The ImageData object holding the image's data (the original image, the recoloured image, and the colour palette).
- **`progress_callback`** (*`QtCore.SignalInstance`*) – Signal that when emitted, is used to update the GUI.

`create_report` ( ) → Optional[colourpaletteextractor.model.generatereport.ColourPaletteReport]

Create a `ColourPaletteReport` object representing the PDF colour palette report.

Returns (Union[ColourPaletteReport, None]) – None if the `ColourPaletteReport` object was not properly generated, otherwise returns the populated `ColourPaletteReport` object.

`save_report` (*pdf: colourpaletteextractor.model.generatereport.ColourPaletteReport*) → None

save the `ColourPaletteReport` object representing the PDF colour palette report to disk.

Parameters **`pdf`** (*`ColourPaletteReport`*) – The `ColourPaletteReport` object to be saved as a PDF to disk..

`colourpaletteextractor.model.generatereport.`**`generate_report`** (*tab: colourpaletteextractor.view.tabview.NewTab*, *image_data: colourpaletteextractor.model.imagedata.ImageData*, *progress_callback: PySide2.QtCore.SignalInstance*) → None

Generate a colour palette report for an image.

Parameters
- **`tab`** (*`NewTab`*) – The tab associated with the image to be analysed.
- **`image_data`** (*`ImageData`*) – The ImageData object holding the image's data (the original image, the recoloured image, and the colour palette).
- **`progress_callback`** (*`QtCore.SignalInstance`*) – Signal that when emitted, is used to update the GUI.

Raises **`ValueError`** – If the provided ImageData object does not have a recoloured image or has no colours in its colour palette.

colourpaletteextractor.model.imagedata module

class `colourpaletteextractor.model.imagedata.`**`ImageData`** (*file_name_and_path: str*)

Bases: `object`

Object to hold the data associated with an image to be analysed.

Stores the original image, its colour palette, the recoloured image, the relative frequency of each colour in the recoloured image, the algorithm used to generate the colour palette and the execution status of the thread used to generate the colour palette.

Parameters **`file_name_and_path`** (*`str`*) – Path to the image to be added.

Raises **`ValueError`** – If the file_name_and_path argument is None.

property `algorithm_used`: type

The algorithm used to generate the image's colour palette.

Returns (type[palettealgorithm.PaletteAlgorithm]) – The class name of the colour palette extraction algorithm.

property `colour_palette`: list

The list of colours in the image's colour palette.

Returns (list[np.array]) – The list of colours ([R,G,B] triplets) in the colour palette.

property `colour_palette_relative_frequency`: list

The relative frequencies of each colour in the colour palette in the recoloured image.

The order of the relative frequencies matches the order of the colours in the colour palette.

Returns (list[float]) – The list of relative frequencies of the colour palette.

property `continue_thread`: bool

Specify if the thread for generating the colour palette or the report should be cancelled.

Returns (bool) – True if the thread should be continued. Otherwise False.

property `extension`: str

The file extension of the original image.

Returns (str) – The file extension of the original image.

property `file_name_and_path`: str

The file path to the original image.

Returns (str) – File path to the original image.

static `get_image_as_q_image` (*image: numpy.array*) → PySide2.QtGui.QImage

Convert a Numpy array representation of an image to a QImage.

Parameters **image** (*np.array*) – An image represented by a Numpy array.

Returns **(QImage)** – The image converted to a QImage.

Raises **ValueError** – If the provided image is not a greyscale, rGB or RGBA image (1, 3, or 4 colour channels).

property image: numpy.array
The original image, represented as a 2 or 3-D Numpy array.

Returns (np.array) – The original image as a Numpy array.

property name: str
The name of the image, without its file extension.

Returns (str) – The image file name, without its extension.

property recoloured_image
The recoloured image, represented as a 3-D Numpy array.

Returns (np.array) – The recoloured image as a Numpy array.

sort_colour_palette (*reverse: bool = True*) → None
Sort the colour palette by their relative frequencies in the recoloured image.

Parameters **reverse** (*bool*) – If True, the colour palette is sorted from largest relative frequency to the smallest. If False, the order is smallest to largest. The default is True.

colourpaletteextractor.model.model module

class colourpaletteextractor.model.model.**ColourPaletteExtractorModel** (*algorithm_class_name=None*)
Bases: object

DEFAULT_ALGORITHM
alias of colourpaletteextractor.model.algorithms.nieves2020.Nieves2020CentredCubes

DEFAULT_HEIGHT: int = 894

DEFAULT_USER_DIRECTORY: str = '/Users/tim/Documents/ColourPaletteExtractor/Output'

DEFAULT_USE_USER_DIRECTORY: bool = False

DEFAULT_WIDTH: int = 1523

ERROR_MSG: str = "Error! :'("

SUPPORTED_IMAGE_TYPES: set = {'jpeg', 'jpg', 'png'}

property active_thread_counter: int

add_image (*file_name_and_path: str*)
From the path to an image, create a new image_data object and add it to the dictionary of image_data objects with a new ID number.

change_output_directory (*use_user_dir: bool, new_user_directory: str*)

close_temporary_directory ( ) → None

evaluate_expression (*expression*)
slot function. :param expression: :return:

generate_palette (*image_data_id, tab, progress_callback=None, temp_algorithm=None*)

get_image_data (*image_data_id*)

property **image_data_id_dictionary**

remove_image_data (*image_data_id*)
Remove image from list of images by its index.

set_algorithm (*algorithm_class_name=<class 'colourpaletteextractor.model.algorithms.nieves2020.Nieves2020CentredCubes'>*)
Set the algorithm use to extract the colour palette of an image.

write_default_settings ( )

colourpaletteextractor.model.model.**generate_colour_palette_from_image** (*path_to_file: str, algorithm: Optional[type] = None*) → tuple

colourpaletteextractor.model.model.**get_settings** ( ) → PySide2.QtCore.QSettings

Module contents

colourpaletteextractor.tests package

Subpackages

colourpaletteextractor.tests.helpers package

Submodules

colourpaletteextractor.tests.helpers.helperfunctions module

colourpaletteextractor.tests.helpers.helperfunctions.**get_image** (*path_to_image: str*)
Returns the image found at the given path.

Parameters **path_to_image** (*str*) – Path to the image to be imported.

Returns **(np.array)** – Image represented as a 3D array

Module contents

Submodules

colourpaletteextractor.tests.nieves2020_test module

colourpaletteextractor.tests.nieves2020_test.**test_closest_relevant_colour_used_to_recol**
( )

colourpaletteextractor.tests.nieves2020_test.**test_cube_colour_must_occur_more_than_thre**
( )

colourpaletteextractor.tests.nieves2020_test.**test_cube_colour_must_occur_more_than_thre**
( )

colourpaletteextractor.tests.nieves2020_test.**test_low_a_b_colour_does_not_meet_secondary**
( )

colourpaletteextractor.tests.nieves2020_test.**test_low_a_b_colour_does_not_meet_secondary**
( )

colourpaletteextractor.tests.nieves2020_test.**test_nieves2020_centred_cubes_constructor**
( )

colourpaletteextractor.tests.nieves2020_test.**test_nieves2020_offset_cubes_constructor**
( )

colourpaletteextractor.tests.nieves2020_test.**test_primary_requirements_1** ( )

colourpaletteextractor.tests.nieves2020_test.**test_primary_requirements_2** ( )

colourpaletteextractor.tests.nieves2020_test.**test_primary_requirements_3** ( )

colourpaletteextractor.tests.nieves2020_test.**test_recoloured_image_of_same_size_1**
( )

colourpaletteextractor.tests.nieves2020_test.**test_recoloured_image_two_colours_1**
( )

colourpaletteextractor.tests.nieves2020_test.**test_recoloured_image_two_colours_2**
( )

colourpaletteextractor.tests.nieves2020_test.**test_two_colours_in_same_cube_can_meet_sec**
( )

Module contents

colourpaletteextractor.view package

Submodules

colourpaletteextractor.view.mainview module

class colourpaletteextractor.view.mainview.**MainView** (*parent=None*)

Bases: PySide2.QtWidgets.QMainWindow

The main window of the ColourPaletteExtractor application.

Parameters parent – Parent object of the MainWindow. Defaults to None.

tabs

tabbed widget for displaying and managing imported images.

Type QTabWidget

colour_palette_dock

Type tabview.ColourPaletteDock

_close_request_action

Action for closing the application

Type QAction

open_action

Action for opening a new image

Type QAction

generate_report_action

Action for generating a report for an image

Type QAction

generate_all_report_action

Action for generating a report for all images with a colour palette

Type QAction

generate_palette_action

Action for generating the colour palette for an image

Type QAction

`generate_all_palette_action`
: Action for generating the colour palette for all images

  Type QAction

`stop_action`
: Action for stopping the report or colour palette being generated for an image

  Type QAction

`preferences_menu_action`
: Action for opening the preferences menu

  Type QAction

`show_help_action`
: Action for showing the quick start guide

  Type QAction

`toggle_recoloured_image_action`
: Action for toggling between the original and the recoloured image

  Type QAction

`zoom_in_action`
: Action for zooming into an image

  Type QAction

`zoom_out_action`
: Action for zooming out of an image

  Type QAction

`about_menu_action`
: Action for showing the about information widget

  Type QAction

`show_palette_dock_action`
: Action for showing the colour palette dock

  Type QAction

`show_toolbar_action`
: Action for showing the toolbar

  Type QAction

`tools`
: (QToolBar): Toolbar for holding QToolButtons used in the GUI

`status`
: Status bar for holding hints, the progress bar and the current version of the application

  Type otherviews.StatusBar

`RESOURCES_DIR` **= 'resources'**
: The name of the directory containing the icons and images used for the GUI.

  Type str

`app_icon` **= 'app_icon'**
: The name of the file used as the application's icon.

Type str

`closeEvent` (*event: PySide2.QtGui.QCloseEvent*) → None
Intercept GUI close event to check if the user wishes to close the GUI.

Parameters `event` (*`QtGui.QCloseEvent`*) – Close event

`close_current_tab` (*tab_index: int*) → int
Close the tab with the given index.

Parameters `tab_index` (*`int`*) – The index of the tab to close

Returns **(int)** – The index of the tab that is now visible after closing the selected tab

`create_new_tab` (*image_id*, *image_data*) → None
Create a new image tab for the main window.

Parameters
- **`image_id`** (*`str`*) – ID of the image to be used for the new tab (e.g., 'Tab_1')
- **`image_data`** (*`model.imagedata.ImageData`*) – Object containing tab and image properties and state

`default_new_tab_image` = **'images:how-to-dark-mode.png'**
The name of the file used as the default new tab (the quick start guide).

Type str

`resources_path` = **'/Users/tim/OneDrive - University of St Andrews/University/MScProject/-ColourPaletteExtractor/colourpaletteextractor/view/resources'**
The path to the resources used for the GUI.
This will vary depending on whether the code has been compiled into an application or is been run from the command line.

Type str

`show_file_dialog_box` (*supported_file_types: set*) → tuple
Show the dialog box for importing images.

Parameters `supported_file_types` (*`set[str]`*) – The supported file types (e.g., '.png')

Returns
- **list** (*str*) – The list of the absolute paths to the images to be loaded into the application
- **str** – The filter used when selecting the images to import

`staticMetaObject` = **<PySide2.QtCore.QMetaObject object>**

colourpaletteextractor.view.otherviews module

class `colourpaletteextractor.view.otherviews.`**`AboutBox`** (*parent=None*)
Bases: `PySide2.QtWidgets.QMessageBox`
Message box to show the basic information about the application.

Parameters `parent` – The parent object of the AboutBox. The default is None.

`staticMetaObject` = **<PySide2.QtCore.QMetaObject object>**

class `colourpaletteextractor.view.otherviews.`**`BatchGenerationProgressWidget`**
Bases: `PySide2.QtWidgets.QDialog`
Custom dialog box shown when multiple colour palette or reports are being generated.
Shows the number of threads to be run and the number of threads completed. Is also has a simple animation attached to it so the user knows that the application has not frozen and is still processing their images.

`label`
Label used to show the number of threads to be run and the number completed.

Type QLabel

`cancel_batch_button`
The button used to notify the controller object that the user wishes to cancel the current batch processing.

Type QPushButton

`set_cancel_text` ( ) → None
Set the text shown to cancelling to let the user know that any incomplete threads are to be cancelled.

`show_widget` (*total_count: int*, *batch_type: str*) → None
Reset and show the widget.

Parameters
- **`total_count`** (*`int`*) – The total number of threads to be processed.
- **`batch_type`** (*`str`*) – The text clarifying what task is being carried out as a batch process.

`staticMetaObject` = <PySide2.QtCore.QMetaObject object>

`update_progress` ( ) → None
Update the batch progress bar by increasing the number of completed threads by one.

class `colourpaletteextractor.view.otherviews.`**`ElidedLabel`** (*text=''*, *width=40*, *parent=None*)
Bases: `PySide2.QtWidgets.QLabel`
Status bar message label that will become elided if there is not enough space to display the entire message.

Adapted from: ref1 and ref2

Accessed: 18/07/2021

Args:
text (str): The text to be shown in the label. The default is an empty string
width (int): The minimum width of the label. The default is 40.
parent: The parent object of the ElidedLabel. The default is None.

`elided_text` ( ) → str
Get the elided text shown by the label.

Returns (str) – The elided text

`paintEvent` (*event: PySide2.QtCore.QEvent.Type.Paint*) → None
Update the text shown by the label on receiving a paint event.

Parameters **`event`** (*`QEvent.Type.Paint`*) – A paint event

`staticMetaObject` = <PySide2.QtCore.QMetaObject object>

class `colourpaletteextractor.view.otherviews.`**`ErrorBox`** (*box_type: Optional[str] = None*, *parent=None*)
Bases: `PySide2.QtWidgets.QMessageBox`
Message box to show warnings and errors.

Parameters
- **`box_type`** (*`str`*) – The error box type. Used to customise the icon and main text show.

- **`parent`** – Parent object of the ErrorBox. Defaults to None.

`header`

The heading of the ErrorBox.

Type str

`append_title` (*error: Exception*) → None

Append the title with additional information from an exception.

Parameters **`error`** (*`Exception`*) – Exception whose error summary message is appended to the title text.

`staticMetaObject` **= <PySide2.QtCore.QMetaObject object>**

class `colourpaletteextractor.view.otherviews.`**`PreferencesWidget`** (*parent=None*)

Bases: `PySide2.QtWidgets.QDialog`

The dialog box for setting a user's preferences.

Currently, the user can change the algorithm used to generate the colour palette, as well as the output directory for any reports that are generated.

Parameters **`parent`** – The parent object of the PreferencesWidget. The default is None.

`browse_button`

Button used to open the operating system's file explorer to select a valid output directory.

Type QPushButton

`user_path_selector`

Text window used to show the user's currently selected output directory.

Type QLineEdit

`default_path_button`

Button used to select the default output directory.

Type QRadioButton

`user_path_button`

Button used to select the user's output directory.

Type QRadioButton

`output_tab`

The output directory settings tab of the preferences dialog box.

Type QWidget

`algorithm_tab`

The algorithm settings tab of the preferences dialog box.

Type QWidget

`get_algorithms_and_buttons` ( ) → tuple

Get the list of algorithm classes and their associated buttons.

Returns

- **(list[palettealgorithm.PaletteAlgorithm])** – List of algorithm classes.
- **(list[QRadioButton]])** – List of buttons associated with the algorithm classes.

`show_output_directory_dialog_box` (*current_path: str*)

Show the dialog box for selecting output directory for reports.

Parameters **`current_path`** (*`str`*) – The path to open the system's file explorer to.

Returns **(str)** – Path to the new output directory.

`show_preferences` ( ) → None
Show the preferences widget.

`staticMetaObject` = <PySide2.QtCore.QMetaObject object>

`update_preferences` ( ) → None
Update the preferences dialog box with the correct settings.

class `colourpaletteextractor.view.otherviews.`**`StatusBar`** (*parent=None*)
Bases: `PySide2.QtWidgets.QStatusBar`
The status bar at the bottom of the main window.
This holds the current shortcut tip for the given tab, as well as the progress bar for showing the current progress towards generating a report or the image's colour palette.

Parameters `parent` – Parent object of the StatusBar. Defaults to None.

`_status_label`
Primary status label.

Type ElidedLabel

`_progress_bar`
Progress bar used to track the progress of generating a colour palette or a report.

Type QProgressBar

`_max_progress`
Maximum value for the progress bar.

Type int

`_min_progress`
Minimum value for the progress bar.

Type int

`set_status_bar` (*state: int*) → None
Set the state of the status bar elements.
Depending on the state, the primary status label will change to reflect what the application is currently processing.

Parameters `state` (*`int`*) – The new state of the status bar.

Raises **`ValueError`** – If state is not a valid state.

`staticMetaObject` = <PySide2.QtCore.QMetaObject object>

`update_progress_bar` (*n: float*) → None
Update the current level of progress for the status bar.

Parameters `n` (*`float`*) – New level of progress for the progress bar.

Raises **`ValueError`** – If the new progress value exceeds the predefined limits of the progress bar.

colourpaletteextractor.view.tabview module

class `colourpaletteextractor.view.tabview.`**`ColourBox`** (*parent=None*)
Bases: `PySide2.QtWidgets.QLabel`
Modified QLabel to hold an individual colour in the colour palette.

Parameters `parent` – The parent object of the ColourBox. The default is None.

enterEvent (*event: PySide2.QtCore.QEvent*) → None

Intercept an enter event.

In the future, this could be used to trigger the highlighting regions of the image that use this colour in the recoloured image.

Parameters event (`QEvent`) – Enter event.

leaveEvent (*event: PySide2.QtCore.QEvent*)

Intercept a leave event.

In the future, this could be used to cancel the highlighting of regions of the image that use this colour in the recoloured image.

Parameters event (`QEvent`) – Leave event.

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class `colourpaletteextractor.view.tabview.`**ColourPaletteDock** (*parent=None*)

Bases: `PySide2.QtWidgets.QDockWidget`

A modified QDockWidget to hold small images of each colour in an image's colour palette.

Parameters parent – Parent object of the ColourPaletteDock. Defaults to None.

add_colour_palette (*colour_palette: list, image_id: str, relative_frequencies: Optional[list] = None*) → None

Clear the colour palette dock and add a new image's colour palette to the dock.

Parameters

- **colour_palette** (`list[np.array]`) – List of colours in the colour palette.
- **image_id** (`str`) – The ID ('Tab_xx') associated with a tab and image.
- **relative_frequencies** (`list[float]`) – The relative frequencies of each colour in the colour palette in the recoloured image.

remove_colour_palette ( ) → None

Remove all of the `ColourBox` labels from the colour palette dock.

Adapted from: ref

Accessed: 27/07/2021

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class `colourpaletteextractor.view.tabview.`**ImageDisplay** (*image_data: colourpaletteextractor.model.imagedata.ImageData, parent=None*)

Bases: `PySide2.QtWidgets.QLabel`

A modified QLabel to display and manipulate the current image.

Parameters

- **image_data** (`imagedata.ImageData`) – The ImageData object that hold the information associated with an image.
- **parent** – Parent object of the ImageDisplay. Defaults to None.

event (*event: PySide2.QtCore.QEvent*) → bool

Intercept the QLabel's event if it is a gesture to allow for zooming into and out of the current image.

Also calls the super class' event handler at the end.

Parameters event (`QEvent`) – An event.

Returns **(bool)** – The result from the super class' event handler.

image_zoom (*mouse_pos: PySide2.QtCore.QPoint, value: float*) → None

Zoom into or out of an image at the mouse pointer's current location.

Parameters
- **`mouse_pos`** (*`QtCore.QPoint`*) – Current position of the mouse cursor.
- **`value`** (*`float`*) – The degree of magnification of the image.

`staticMetaObject` **= <PySide2.QtCore.QMetaObject object>**

`update_image` (*image: numpy.array*) → None

Update the image shown by the ImageDisplay.

Parameters **`image`** (*`np.array`*) – Numpy array representing an image.

`zoom_factor` **= 1.25**

The zoom-in factor used when the user zoom's into the image via the zoom-in button.

`zoom_in` (*zoom_factor: float = 1.25*) → None

Zoom into the current image.

Parameters **`zoom_factor`** (*`float`*) – The new magnification factor for the image.

`zoom_out` (*zoom_factor: float = 0.8*) → None

Zoom out of the current image.

Parameters **`zoom_factor`** (*`float`*) – The new magnification factor for the image.

`zoom_out_factor` **= 0.8**

The zoom-out factor used when the user zoom's out of the image via the zoom-out button.

class `colourpaletteextractor.view.tabview.`**`NewTab`** (*image_id: Optional[str] = None, image_data: Optional[colourpaletteextractor.model.imagedata.ImageData] = None, parent=None*)

Bases: `PySide2.QtWidgets.QScrollArea`

Modified QScrollArea to display and manipulate an image (via the `ImageDisplay` class).

Parameters
- **`image_id`** (*`str`*) – The ID ('Tab_xx') associated with a tab and image.
- **`image_data`** (*`imagedata.ImageData`*) – The ImageData object that hold the information associated with an image.
- **`parent`** – Parent object of the NewTab Defaults to None.

`image_display`

ImageDisplay used to show the QPixmap representation of the current image.

Type ImageDisplay

`change_toggle_recoloured_image_pressed` ( ) → None

Toggle the _toggle_recoloured_image_pressed attribute between true and false (its opposite).

property `generate_palette_available`: bool

The ability to generate the colour palette for the current NewTab object.

Returns (bool) – Returns true if the colour palette can be generated. Otherwise false.

property `generate_report_available`: bool

The ability to generate the colour palette report for the current NewTab object.

Returns (bool) – Returns true if the colour palette report can be generated. Otherwise false.

`get_slider_positions` ( ) → PySide2.QtCore.QPointF

Get the grip positions of the horizontal and vertical scrollbars.

Returns (QPointF) – The position of the grip for the horizontal and veritcal scrollbars.

property `image_id`: str
The image ID of the images and its data that is linked to the current NewTab object

Returns (str) – The ID ('Tab_xx') associated with a tab and image.

property `progress_bar_value`: float
The current level of progress shown by the status bar for the associated NewTab object.

Returns (float) – The current level of progress shown by thr status bar.

`set_slider_positions` (*x_position: float*, *y_position: float*) → None
Set the position of the horizontal and vertical scrollbar's grip.

Parameters
- **`x_position`** (*`float`*) – Position of the grip for the horizontal scrollbar.
- **`y_position`** (*`float`*) – Position of the grip for the vertical scrollbar.

`staticMetaObject` = <PySide2.QtCore.QMetaObject object>

property `status_bar_state`: int
The current status bar state, represented by an integer.
See the `otherviews.StatusBar.set_status_bar()` method for more information.

Returns (int) – The current status bar state.

property `toggle_recoloured_image_available`: bool
Stores the availability of the recoloured image (if it available to be displayed or not).

Returns (bool) – True if the recoloured image is available. Otherwise false.

property `toggle_recoloured_image_pressed`: bool
The status of the toggle button used to switch between the original image and the recoloured image.

Returns (bool) – True if the recoloured image is displayed by the GUI. Otherwise false.

`wheelEvent` (*event: PySide2.QtGui.QWheelEvent*) → None
Intercepts the super class' wheelEvent to allow zooming into and out of an image using the mousewheel.
Also calls the super class' wheelEvent handler at the end.

Parameters **`event`** (*`QWheelEvent`*) – Mousewheel event

property `zoom_level`: float
The degree of magnification for the currently displayed image.

Returns (float) – The degree of magnification for the current image.

Module contents

1.1.2 Module contents

- genindex
- modindex
- search