npGUI

Release 0.1.0

Gaël Cousin

May 15, 2023

CONTENTS

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

1.1 The np_gui package

This package implements Graphical User Interfaces from a NumPy user perspective: every image corresponds to an ndarray.

The core class is ClickableImage whose instances' attributes are:

- a dictionary of parameter values,
- a callable that returns an image of fixed shape,
- regions in that shape defined as boolean images of the same shape
- and their corresponding callbacks.

When calling the use method of an instance, the dictionary is passed to the callable, that uses (some of) its values as arguments and outputs an image which is displayed in a window.

Upon clicking on the distinguished regions, their callbacks are called and alter the parameter values. The image is then refreshed from this input.

When closing the window, the use method returns the dictionary with its altered values; whence the interaction with the user.

These objects can be stacked vertically and horizontally to compose more complex ones. A certain number of subclasses are proposed as basic building blocks. The stacked blocks may have parameter keywords in common, which allows for interactions between them.

The code is organized in three submodules as follows.

- **np_clickable_images** contains the definition of the ClickableImage class and some subclasses,
- **colors** contains some color management facilities and
- **image_annotation** regards putting text in an image.

1.2 The np_clickable_image module

This module implements ClickableImage and some subclasses.

Classes:

_CombinableFunc	A class of callables mimicking positional keywords functions.
ClickableImage	A class for clickable images, made with matplotlib and numpy.
ImageToggles	A subclass of ClickableImage where regions of the image become toggles.
RadioButton	A subclass for radio buttons.
SliceDisplayer	This subclass is for clickable images that display a slice of text.
Button	The subclass of ClickableImage made of basic rectangular buttons.

`class np_gui.np_clickable_image._CombinableFunc`

A class of callables mimicking positional keywords functions.

A class of callables mimicking functions with only positional keyword arguments with better control on signature and default values. This class does not care about the specific order of the positional keyword arguments.

Methods:

defaults_dictionary(f)	Parse the keyword(-only) arguments default values.
__init__(func[, defaults_dic])	Build the callable.

`static defaults_dictionary(f)`

Parse the keyword(-only) arguments default values.

Parameters

f (*Callable[Any,Any]*) – A function.

Returns

A dictionary with keys given by the keyword arguments
and keyword-only arguments of f that maps these keys to the corresponding default values.

Return type

dict

__init__(*func*, *defaults_dic=None*)

Build the callable.

Build the callable from a keyword(-only) function and optionally a dictionary. The function can be defined with a signature (**kwargs, *, …). In this case, the placeholder for a detailed specification of keyword arguments is the dictionary, see below. If the dictionary is not passed, the function's signature must specify every admissible positional keyword argument. Functions using also keyword-only arguments can be passed but the keyword-only arguments with no default value will be ignored. If a dictionary is passed, the function's signature is ignored.

Parameters

- **func** (*Callable*) – A function with solely keyword and keyword-only arguments;

- **dic** (*dict, optional*) – A dictionary specifying all admissible positional keyword arguments TOGETHER WITH their default values. Defaults to None.

class np_gui.np_clickable_image.**ClickableImage**

A class for clickable images, made with matplotlib and numpy.

Methods:

__init__(image, shape, regions, callbacks, ...)	Builder method
use([return_vars])	Use the clickable image to get some user input.
constant_to_clickable(img)	Convert np.ndarray image to ClickableImage.
hstack(clickables)	Stack horizontally the input ClickableImages or np.ndarray images.
vstack(clickables)	Stack vertically the input ClickableImages or np.ndarray images.
resize(shape[, anti_aliasing, preserve_range])	Resize self to input shape.
center_in_shape(shape[, frame_color])	Return clickable of the given shape with self in the center.

__init__(*image*, *shape*, *regions*, *callbacks*, *vars_dic*)

Builder method

Parameters

- **image** (*Callable[Any, np.ndarray] | np.ndarray*) – A function that returns an image or a 'constant' image. If a function is passed, it must have only keyword and keyword-only arguments. The keyword-only arguments with no default value will be ignored.
- **shape** (*tuple[int, ...]*) – shape of the ndarray returned by image
- **regions** (*list[np.ndarray]*) – boolean arrays A with A.shape=shape[:2]
- **callbacks** (*list[Callable[dict, None]]*) – functions to be called when the regions are clicked (the ith callback belongs to the ith region)
- **vars_dic** (*dict*) – some variable values stored in a dict. They are meant to define the behaviour of the image attributes of the current instance and possibly other ones, in view of interactions between instances (stacking).

Raises

- **ValueError** – The number of regions does not equal the number of callbacks.
- **ValueError** – The regions' shapes do not all equal shape[:2]
- **ValueError** – The passed regions do not have dtype('bool') dtype.

use(*return_vars=None*, ***plot_options*)

Use the clickable image to get some user input.

For this method to work, all the keyword arguments for image must be contained in vars_dic.

Parameters

- **return_vars** (*list[str] | None, optional*) – The keys of vars_dic we are interested in. Defaults to None. If None is received, all vars_dic will be returned after the interaction.

- `**plot_options` – the optional arguments to be passed to imshow. Useful to affect the display.

Returns
The requested dictionary of variables values.

Return type
dict

`static constant_to_clickable`(*img*)

Convert np.ndarray image to ClickableImage.

Parameters
`img` (*`np.ndarray`*) – an image

Returns
The corresponding “constant” ClickableImage.

Return type
ClickableImage

`static hstack`(*clickables*)

Stack horizontally the input ClickableImages or np.ndarray images.

Parameters
`clickables` (*`list[ClickableImage | np.ndarray]`*) – list of ClickableImages or images, in the second case they are treated as ‘constant’ ClickableImages.

Raises
`ValueError` – The clickables are not stackable as demanded, due to shape issues.

Returns

The ClickableImage obtained by stacking
horizontally the input clickables/images, with possible interactions if they share keys of their image.defaults_dic | vars_dic

Return type
ClickableImage

`static vstack`(*clickables*)

Stack vertically the input ClickableImages or np.ndarray images.

Parameters
`clickables` (*`list[ClickableImage | np.ndarray]`*) – list of ClickableImages or images, in the second case they are treated as’constant’ ClickableImages.

Raises
`ValueError` – The clickables are not stackable as demanded, due to shape issues. Remember this could include a depth issue.

Returns

The ClickableImage obtained by stacking
vertically the input clickables, with possible interactions if they share keys of their image.defaults_dic | vars_dic

Return type
ClickableImage

`resize`(*shape*, *anti_aliasing=False*, *preserve_range=False*)

Resize self to input shape.

Parameters

- **shape** (*tuple[int,int]*) – Desired 2d shape for the output.
- **anti_aliasing** (*bool, optional*) – argument to be passed in the underlying skimage.resize call. Defaults to False.
- **preserve_range** (*bool, optional*) – argument to be passed in the underlying skimage.resize call. Defaults to False.

Returns

The resized version of self, with the same
number of channels as self.

Return type
ClickableImage

center_in_shape(*shape*, *frame_color=None*)

Return clickable of the given shape with self in the center.

Parameters

- **shape** (*tuple[int, int]*) – the 2d shape in which self should be centered
- **frame_color** (*_type_, optional*) – The color of the added pixels.
- **passed** (*Defaults to None. If None is*) – self.get_image() will be used. Otherwise, this argument must be parsable by np_gui.colors.color_to_rgb.
- **of** (*the main color*) – self.get_image() will be used. Otherwise, this argument must be parsable by np_gui.colors.color_to_rgb.

Returns

A new clickable A with A.shape=shape,
self in the center and extra pixels color specified by frame_color.

Return type
ClickableImage

class np_gui.np_clickable_image.**ImageToggles**

A subclass of ClickableImage where regions of the image become toggles.

No new method, a unique original method: __init__ which 'extends' the superclass's method.

Methods:

__init__(image, regions, *[, daltonism, ...])	Build self.

__init__(*image*, *regions*, *, *daltonism=False*, *on_color='green'*, *off_color='red'*, *default_toggle_value=True*, *true_is_black_pixel=True*, *toggles_name='toggles'*)

Build self. Extend super().__init__, with distinct arguments.

Parameters

- **image** (*np.ndarray*) – The "background" image
- **regions** (*list[np.ndarray]*) – the 'toggle' regions as boolean images.
- **daltonism** (*bool, optional*) – Option to set a special color mode for color blind people. Defaults to False.

- **on_color** (*str, optional*) – The color of the regions when toggled to True. Defaults to "green". This string must be recognized by matplotlib.colors.to_rgb.
- **off_color** (*str, optional*) – The color of the regions when toggled to False. Defaults to "red". This string must be recognized by matplotlib.colors.to_rgb.
- **default_toggle_value** (*bool, optional*) – The common value of all
- **True.** (*toggles as defined in the Clickable image vars_dic. Defaults to*) –
- **true_is_black_pixel** (*bool, optional*) – Decide if the front pixel are black, in case image is boolean. Defaults to True.
- **toggles_name** (*str, optional*) – the key of the toggles values list in the ClickableImage's vars_dic. Defaults to "toggles".

Raises

`TypeError` – "The input regions do not have 'bool' dtype.")

class np_gui.np_clickable_image.**RadioButton**

A subclass for radio buttons.

It has a unique original method: the _init_ method extends the superclass' one.

Methods:

__init__(radius[, toggle_name, checked])	Instantiate RadioButton.

__init__(*radius*, *toggle_name='radio_toggle'*, *checked=False*)

Instantiate RadioButton.

Parameters

- **radius** (*int*) – radius of the external circle of the button.
- **toggle_name** (*str, optional*) – vars_dic keyword for the button status (checked or not). Defaults to "radio_toggle".
- **checked** (*bool, optional*) – Initial state of the button. Defaults to False. If False, the button starts unchecked. Otherwise, it starts checked.

class np_gui.np_clickable_image.**SliceDisplayer**

This subclass is for clickable images that display a slice of text.

The slice may have variable start and stop.

Methods:

__init__(text, shape, *[, background_color, ...])	Extends __init__ of superclass, with distinct arguments.

__init__(*text*, *shape*, *, *background_color='white'*, *slice_varname='slice'*)

Extends __init__ of superclass, with distinct arguments.

Parameters

- **text** (*str*) – The text whose slice will be displayed
- **shape** (*tuple[int, int]*) – The shape of the resulting ClickableImage

- **background_color** (*str, optional*) – Background color, this string must be recognized by matplotlib.colors.to_rgb. Defaults to "white".
- **slice_varname** (*str, optional*) – The name of the variable that defines the slice to take. Defaults to "slice".

`class np_gui.np_clickable_image.Button`

The subclass of ClickableImage made of basic rectangular buttons.

More precisely, the unique clickable region is the whole image.

Methods:

__init__(image, callback)	Button constructor.

__init__(*image*, *callback*)

Button constructor.

Parameters

- **image** (*np.ndarray*) – The underlying image of the button.
- **callback** (*Callable*) – The callback triggered by clicking the button.

1.3 The colors module

Provide a few color conversion, color picking facilities.

The reference color format here is the one of a numpy array with shape (3,) and dtype np.uint8. Beware that this is not the same convention as in matplotlib.colors. In this package, interacting with this package, it is recommended to always prefer our colors.color_to_rgb to matplotlib.colors.to_rgb.

Otherwise, you could run in incompatibily issues.

Functions:

color_to_rgb(color)	Convert matplotlib color to 8-bits rgb color.
binary2grayscale(image)	Convert a binary image to rgb image.
grayscale2rgb(image)	Convert a grayscale image to rgb image.
binary2rgb(image)	Convert a binary image to rgb image.
to_rgb(image)	Convert image to the rgb format.
main_color(image[, region])	Return the most frequent color of image within region.
mono_block(shape, color)	Return rgb image of the given shape and color.

`np_gui.colors.`**`color_to_rgb`**(*color*)

Convert matplotlib color to 8-bits rgb color.

Parameters

color (*tuple[int, int, int] | list[int] | tuple[float, float, float] | list[float] | str | np.ndarray*) – an 8-bits rgb color or its float rescaled version, with values in [0,1] or a string recognized by matplotlib.to_rgb.

Raises

- **ValueError** – "Expecting a 3-channels color (rgb) or a string."

- `ValueError` – "To specify your color by a triple of integers, they should belong to the closed interval [0,255]."

Returns

an np.ndarray of shape (3,) and dtype 'uint8', representing the rgb color.

Return type
np.ndarray

`np_gui.colors.`**`binary2grayscale`**`(`*image*`)`

Convert a binary image to rgb image.

Parameters
`image` (*`np.ndarray`*) – A binary image as a 2d np.ndarray

Returns
the corresponding grayscale image.

Return type
np.ndarray

`np_gui.colors.`**`grayscale2rgb`**`(`*image*`)`

Convert a grayscale image to rgb image.

Parameters
`image` (*`np.ndarray`*) – A grayscale image as a 2d np.ndarray with dtype 'uint8'.

Returns
the corresponding rgb image.

Return type
np.ndarray

`np_gui.colors.`**`binary2rgb`**`(`*image*`)`

Convert a binary image to rgb image.

Parameters
`image` (*`np.ndarray`*) – A binary image as a 2d np.ndarray

Returns
the corresponding rgb image

Return type
np.ndarray

`np_gui.colors.`**`to_rgb`**`(`*image*`)`

Convert image to the rgb format.

Parameters
`image` (*`np.ndarray`*) – The image to be converted.

Raises
`ValueError` – The expected dtype of the ndarray is either 'bool' or 'uint8'. The expected shape is either 2d or 3d with shape[2]==3.

Returns
The image converted to the rgb format.

Return type
np.ndarray

`np_gui.colors.`**`main_color`**`(`*image*, *region=None*`)`

Return the most frequent color of image within region.

Parameters

- **`image`** (*`np.ndarray`*) – The 2d image to be studied as 2d or 3d np.ndarray.
- **`region`** (*`np.ndarray | None , optional`*) – The region in which the frequency should be calculated, as a boolean 2d image. Defaults to None. if None is passed, the full image is considered.

Raises

`ValueError` – "region has no front pixel!"

Returns

The most frequent color in image within region, as an np.ndarray (most likely a 1,3 or 4 entries 1d array)

Return type

np.ndarray

`np_gui.colors.`**`mono_block`**`(`*shape*, *color*`)`

Return rgb image of the given shape and color.

Parameters

- **`shape`** (*`tuple[int,int]`*) – The 2d shape of the sought block
- **`color`** – A color, as accepable by color_to_rgb

Returns

The sought monochrome image as an rgb np.ndarray.

Return type

np.ndarray

1.4 The image_annotation module

This module deals with displaying text on a Numpy image.

Functions:

`string_to_image`(s[, relative_line_spacing])	Produce image of the given string.
`center_text`(text, shape[, color, ...])	Center given text over a monochrome background to create a np.ndarray.

`np_gui.image_annotation.`**`string_to_image`**`(`*s*, *relative_line_spacing=0.1*`)`

Produce image of the given string.

Parameters

- **`s`** (*`str`*) – The string that must appear on the image.
- **`relative_line_spacing`** (*`float, optional`*) – The line spacing, relative to the height of ONE line. Defaults to 0.1.

Returns

boolean image displaying the input string. The text corresponding to null pixels.

Return type
np.ndarray

`np_gui.image_annotation.`**`center_text`**(*text*, *shape*, *color='black'*, *background_color='white'*)

Center given text over a monochrome background to create a np.ndarray.

Parameters

- **`text`** (*`str`*) – The text to print
- **`shape`** (*`tuple[int,int]`*) – The shape of the background
- **`color`** – The color of the background, of a type acceptable by colors.color_to_rgb. Defaults to black.
- **`background_color`** – The color of the background, of a type acceptable by colors.color_to_rgb. Defaults to white.

Returns

The resulting image as an rgb np.ndarray of dtype 'uint8'.
The text fills 90% of the width or of the height of the input shape, depending on its own proportions.

Return type
np.ndarray

PYTHON MODULE INDEX

INDEX