image registration

Release 1.0.4

Fabian Preiss

Jun 01, 2021

TABLE OF CONTENTS

This code implements an image processing library intended for batch processing of image registration algorithms.

CHAPTER

ONE

UTIL

graph	Convenience functions for directed acyclic graphs
methods	Convenience functions for registration related image methods
params	Parameter classes implementing lazy evaluation and dependency resolution.
solver	Solvermodel primitives.
helpers	Collection of useful helper functions in specific use-cases
io	Directory processing utilities

1.1 graph

Convenience functions for directed acyclic graphs

Author: Fabian A. Preiss.

Classes

DAGraph(vertex_parent_dict, Set[Hashable]])	Collection of functions for directed acyclic graphs.

1.1.1 DAGraph

class imgreg.util.graph.**DAGraph**(*vertex_parent_dict: Dict[Hashable, Set[Hashable]]*, *invert=False*)

Bases: object

Collection of functions for directed acyclic graphs.

This class implements functions for directed acyclic graphs[1][2]. The graph is stored in a dictionary, where the keys define the identifiers for the vertices and the value contains the set of parents. This implementation is motivated for handling generic problems where dependencies appear (see dependency graphs[3]).

[1] Wikipedia, "Graph theory"
[2] Wikipedia, "Directed acyclic graph"
[3] Wikipedia, "Dependency graph"

This module is for experimental use only, more extensive graph libraries for python are available under[5][6][7][8].

Parameters

vertex_parent_dict [dict[str, set[Hashable]]] A dictionary mapping each vertex to all of its parents. It is assumed, that the input is a directed acyclic graph (connected or disconnected). Cycle Detection is not performed on the input.

invert [boolean] invert the edges of the input graph

Notes

A directed graph is an ordered pair $G = (V, E)$, where

- V is a set of vertices (also nodes or points)
- $E \subseteq \{(x, y) \mid (x, y) \in V^2 \text{ and } x \neq y\}$ is a set of ordered pairs of vertices called edges.

To represent a graph, DAGraph takes a single python dictionary as input, where `key`, $parent \in V$ and

$$\texttt{value} = pa\,(\texttt{key}) = \{\{parent\} \mid (\texttt{key}, parent) \in E\}$$

References

Methods

`__init__`(vertex_parent_dict[, invert])	Initialize self.
`ascendants`(vertex)	Given vertex, return a set containing all its ascendants[4].
`children`(vertex[, order])	Given vertex, return a set containing all its children.
`descendants`(vertex)	Given vertex, return a set containing all descendants[?].
`parents`(vertex[, order])	Given vertex, return a set containing all its parents.

Attributes

`vertex_ascendants_dict`	Get a dictionary mapping each vertex to all of its ascendants[?].
`vertex_parent_dict`	Get a dictionary mapping each vertex to all of its parents.

`ascendants`(*vertex: Hashable*) → Set[Hashable]
Given vertex, return a set containing all its ascendants[?].

Returns

Set Set of *ascendants*

`children`(*vertex: Hashable*, *order: int = 1*) → Set[Hashable]

[5] https://networkx.org/
[6] https://graph-tool.skewed.de/
[7] https://pygsp.readthedocs.io/en/stable/
[8] https://igraph.org/python/
[4] Wikipedia, "Tree (graph theory)"

Given vertex, return a set containing all its children.

Returns

Set Set of *children*

descendants(*vertex: Hashable*) → Set[Hashable]
Given vertex, return a set containing all descendants[?].

Returns

Set Set of *descendants*

parents(*vertex: Hashable*, *order=1*) → Set[Hashable]
Given vertex, return a set containing all its parents.

Returns

Set Set of *parent*

property vertex_ascendants_dict
Get a dictionary mapping each vertex to all of its ascendants[?].

Returns

dict vertex : Set of vertices

property vertex_parent_dict
Get a dictionary mapping each vertex to all of its parents.

Returns

dict vertex : Set of vertices

1.2 methods

Convenience functions for registration related image methods

Author: Fabian A. Preiss.

Classes

ImageMethods()	Collection of static methods for image analysis and manipulation.

1.2.1 ImageMethods

class imgreg.util.methods.**ImageMethods**
Bases: object

Collection of static methods for image analysis and manipulation.

Methods

`__init__`(*args, **kwargs)	Initialize self.
abs_diff(image, ref_image)	Absolute value of the difference between two images.
compute_afts(image)	Compute FFT magnitude, shifted with low fequencies in center.
compute_dgfw(image[, gaussdiff, ...])	Image Difference of Gaussian Filter + Window.
compute_log_polar_tf(image[, wrexp, order])	Compute log-scaled polar coordinate transform of center shifted FFT.
compute_rts(image[, angle, scale, ...])	Rotate, translate and scale image.
compute_warp_radius(image_diameter[, wrexp])	Compute the warp radius from image and warp radius exponent *wrexp*.
exp_filter(image[, signal_noise_ratio])	Remap the values of the image such that bright pixels are given an exponentially higher weight.
max_sinogram_angle(image[, theta, ...])	Iterative solver, works well in special cases.
norm_rel_l2(image, ref_image)	Compute a relative similarity measurement between two images.
recover_rs(image_warped_fs, rts_warped_fs, ...)	Recover the rotation and scaling transformation from given input.
sinogram(image[, theta, exp_filter_val, circle])	Computes the radon transformation and optionally applies the *exp_filter* afterwards.
sinogram_project(image[, theta, ...])	Projects the sinogram onto the axis of the theta angle.
sqr_diff(image, ref_image)	Squared difference between two images.

static abs_diff(*image: numpy.ndarray*, *ref_image: numpy.ndarray*) → numpy.ndarray
Absolute value of the difference between two images.

Notes

$$\texttt{abs}\left(\texttt{ref\_image} - \texttt{image}\right)$$

static compute_afts(*image: numpy.ndarray*) → numpy.ndarray
Compute FFT magnitude, shifted with low fequencies in center.

Parameters

image [numpy.ndarray] The input image for the fourier transform

Returns

numpy.ndarray FFT magnitude, center shifted

static compute_dgfw(*image: numpy.ndarray*, *gaussdiff: Sequence[float] = (5, 20)*, *windowweight: float = 1*, *windowtype: str = 'hann'*) → numpy.ndarray
Image Difference of Gaussian Filter + Window.

Parameters

image [numpy.ndarray] The input image to filter

gaussdiff [(float, float), optional] The low and high standard deviations for the gaussian difference band pass filter

windowweight [float, optional] weighting factor scaling beween windowed image and image

windowtype [str, optional] see *skimage.filters.window* for possible choices

Returns

numpy.ndarray modified image with bandpass filter and window applied

Notes

Applying this bandpass and window filter prevents artifacts from image boundaries and noise from contributing significantly to the fourier transform. The gaussian difference filter can be tuned such that the features relevant for the identification of the rotation angle are at the center of the band pass filter.

static compute_log_polar_tf(*image: numpy.ndarray*, *wrexp: float = 3*, *order: int = 5*) → numpy.ndarray

Compute log-scaled polar coordinate transform of center shifted FFT.

Parameters

image [numpy.ndarray] The input image to transform (expects center shifted FFT magnitude)

wrexp [float, optional] Cutoff exponent factor for higher frequencies, larger wrexp => faster computation min value: 1

Returns

numpy.ndarray log-scaled polar transformed of the input image

static compute_rts(*image: numpy.ndarray*, *angle: float = 0*, *scale: float = 1*, *translation: Sequence[float] = (0.0, 0.0)*, *inverse: bool = False*, *preserve_range: bool = True*, *order: int = 5*) → numpy.ndarray

Rotate, translate and scale image.

Parameters

image [numpy.ndarray] The input image to transform

angle [float, optional] The rotation angle in degrees for the transform

scale [float, optional] The scaling factor used in the transform

translation [(float, float), optional] x,y-translations used in the transform

inverse [bool] Apply the backwards transformation for given parameters

Returns

numpy.ndarray modified image with same shape as initial image

static compute_warp_radius(*image_diameter: int*, *wrexp: float = 1.0*) → int

Compute the warp radius from image and warp radius exponent *wrexp*.

Parameters

image_diameter [int] The length of the smallest image dimension

wrexp [float, optional] Cutoff exponent factor for higher frequencies, larger wrexp => faster computation min value: 1

Returns

int The cutoff radius for the log-ploar transform of the image

static exp_filter(*image: numpy.ndarray*, *signal_noise_ratio: Optional[float] = None*) → numpy.ndarray

Remap the values of the image such that bright pixels are given an exponentially higher weight.

Maps the min value of the input image to 1/signal_noise_ratio and the max value to 1.

static max_sinogram_angle(*image*, *theta=None*, *exp_filter_val=None*, *circle=False*, *precision=0.1*) → float

Iterative solver, works well in special cases. Implementation very crude.

static norm_rel_l2(*image: numpy.ndarray*, *ref_image: numpy.ndarray*) → float

Compute a relative similarity measurement between two images.

Interpretes the images as a vector and calculates the L2 norm of the differences relative to the reference image *ref_image*.

Notes

L2 norm Implemented analog to *NormRel_L2*[1].

$$\frac{\|\texttt{ref\_image} - \texttt{image}\|_F}{\|\texttt{ref\_image}\|_F}$$

Where $\|\ldots\|_F$ denotes the Frobenius norm of a matrix.

References

static recover_rs(*image_warped_fs: numpy.ndarray*, *rts_warped_fs: numpy.ndarray*, *image_shape: Sequence[int]*, *upsampl: int = 10*, *wrexp: float = 3*) → Tuple[numpy.ndarray, numpy.ndarray, Dict[str, Hashable]]

Recover the rotation and scaling transformation from given input.

Parameters

image_warped_fs [np.ndarray] log-polar warped fourier transformed of original input image

rts_warped_fs [np.ndarray] log-polar warped fourier transformed of modified input image

image_shape [Sequence[int]] image dimensions of original input image

upsampl [int, optional] Upsampling factor. 1 => no upsampling, 20 => precision to 1/20 of a pixel

wrexp [float, optional] Cutoff exponent factor for higher frequencies, larger wrexp => faster computation min value: 1

Returns

numpy.ndarray Vector of recovered rotation angle and error in degrees

numpy.ndarray Vector of recovered scaling factor and error

dict Dict containing the phase_cross_correlation parameters

[1] NVIDIA Performance Primitives (NPP) - Image Norms

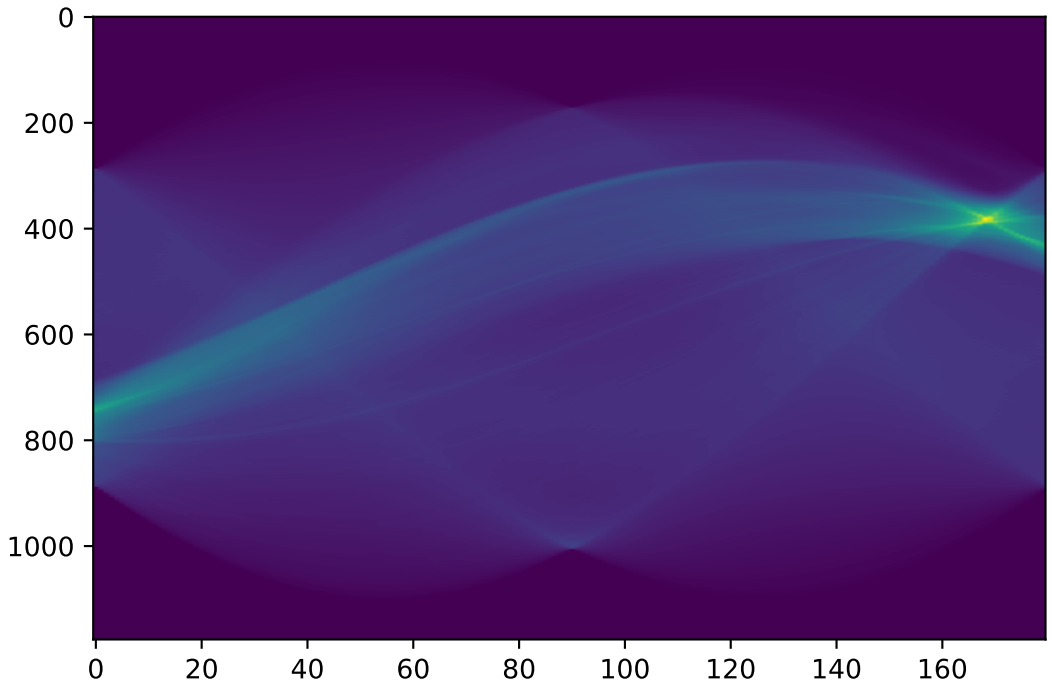
Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

static sinogram(*image*, *theta=None*, *exp_filter_val=None*, *circle=False*) → numpy.ndarray
Computes the radon transformation and optionally applies the *exp_filter* afterwards.

Examples

```
import numpy as np
import matplotlib.pyplot as plt
import imgreg.data as data
from imgreg.util.methods import ImageMethods

img = np.array(data.mod_img())

# Compute the sinogram using the radon transform
sinogram = ImageMethods.sinogram(img)
plt.imshow(sinogram, aspect=0.1)
plt.show()
```

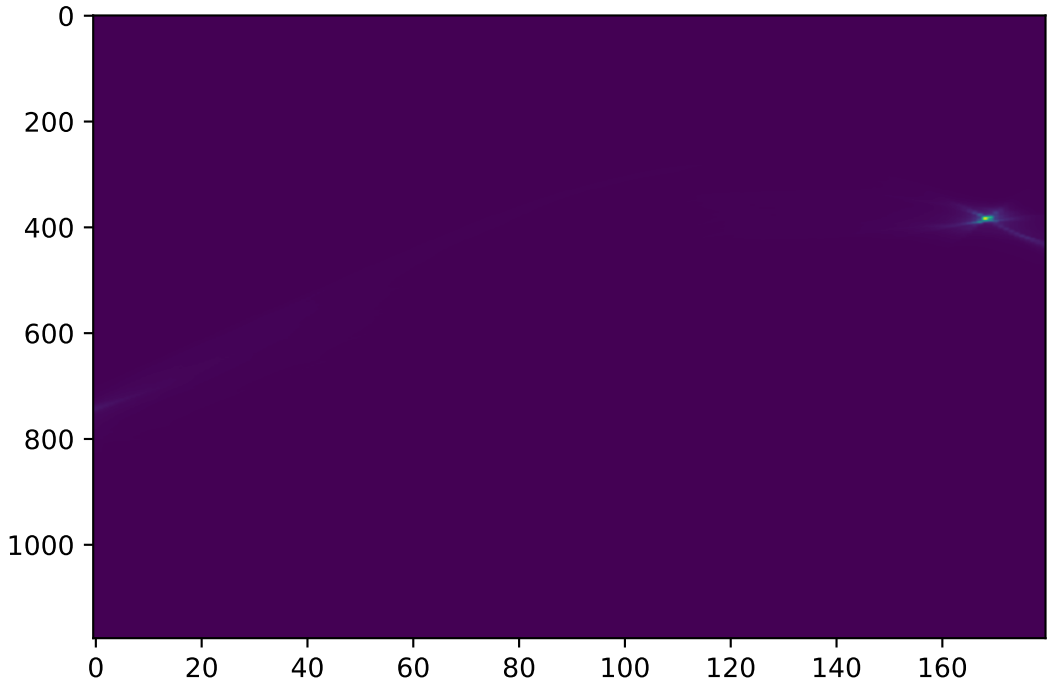



```
# Compute the same sinogram but apply an exponential weighting filter
sinogram = ImageMethods.sinogram(img, exp_filter_val=1000)
```

(continues on next page)

(continued from previous page)

```
plt.imshow(sinogram, aspect=0.1)
plt.show()
```

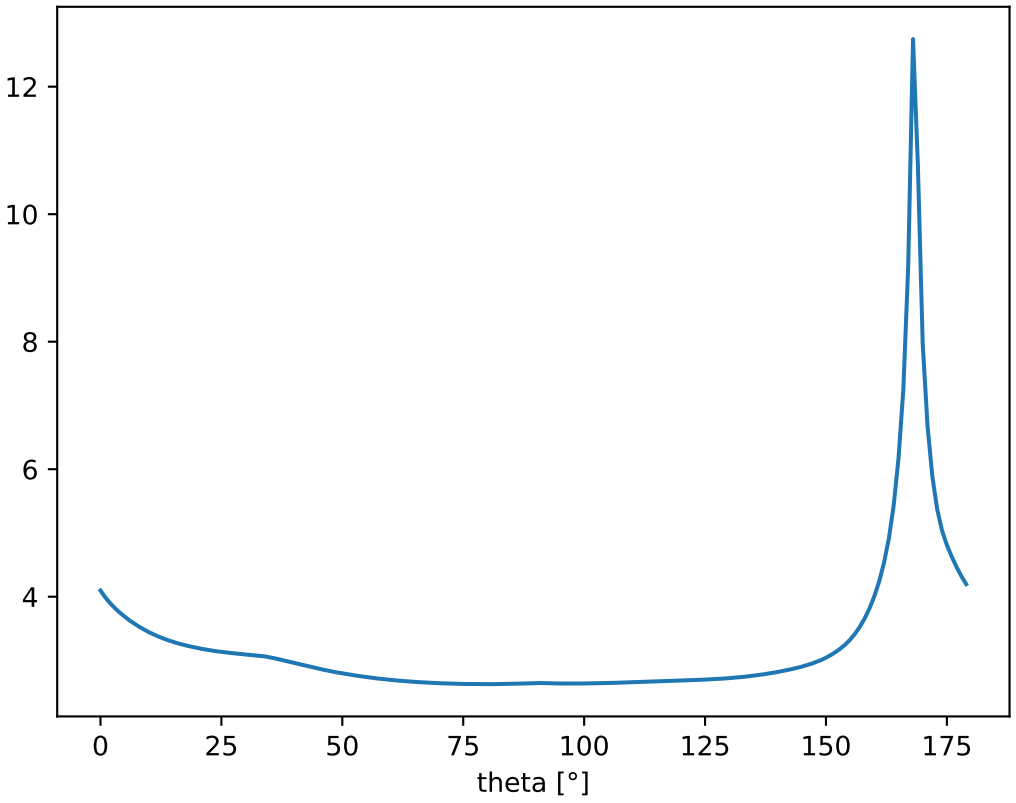


static sinogram_project(*image*, *theta=None*, *exp_filter_val=None*, *circle=False*) → numpy.ndarray

Projects the sinogram onto the axis of the theta angle.

Examples

```
import numpy as np
import matplotlib.pyplot as plt
import imgreg.data as data
from imgreg.util.methods import ImageMethods

img = np.array(data.mod_img())

# Compute the sinogram with the exponential weighting filter and project the␣
↪image values
# to the axis corresponding to the theta angles.
sinogram_project = ImageMethods.sinogram_project(img, exp_filter_val=1000)
plt.plot(sinogram_project)
plt.xlabel("theta [°]")
plt.show()
```


12
10
8
6
4
0
25
50
75
100
125
150
175
theta [°]

static sqr_diff(*image: numpy.ndarray*, *ref_image: numpy.ndarray*) → numpy.ndarray
Squared difference between two images.

Notes

$$(\texttt{ref\_image} - \texttt{image})^2$$

1.3 params

Parameter classes implementing lazy evaluation and dependency resolution.

Author: Fabian A. Preiss

Functions

interface_function_handle(parameter)	Construct value using parent *Parameter*s

1.3.1 interface_function_handle

imgreg.util.params.**interface_function_handle**(*parameter: Dict[enum.Enum, imgreg.util.params.Parameter]*) → Any
Construct value using parent *Parameter*s

Classes

ImageParameter(enum_id, value_definition, ...)	*Parameter* with additional display functionality for stored image.
Parameter(enum_id, value_definition, ...)	Base *Parameter* class, aware of related *Parameter*s and lazy evaluating.

1.3.2 ImageParameter

class imgreg.util.params.**ImageParameter**(*enum_id: enum.Enum*, *value_definition: Union[Type[Any], Callable[[Dict[enum.Enum, imgreg.util.params.Parameter]], Any]]*, *parent_parameters: Optional[Set[imgreg.util.params.Parameter]] = None*)
Bases: *imgreg.util.params.Parameter*

Parameter with additional display functionality for stored image.

Methods

__init__(enum_id, value_definition[, ...])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

add_child(*child:* imgreg.util.params.Parameter) → None
Assign a Parameter a child relation

add_descendant(*descendant:* imgreg.util.params.Parameter) → None
Assign a Parameter a child relation

property aspect
The aspect ratio used by matplotlib.

property bounds
Cropping boundaries for the display function.

property children
*Parameter*s with a child relation to this *Parameter*s instance.

clear(*clear_descendants=True*) → None
Clear the value stored in this and dependent *Parameter*s.

property cmap
The colormap used by matplotlib.

property const
Flag if the value can be overwritten once initialized.

property descendants
*Parameter*s with a descendant relation to this *Parameter*s instance.

display(*axsp: Optional[Any] = None*)
Display the stored image using matplotlib.

Parameters

aspx [matplotlib.axes._subplots.AxesSubplot] AxesSubplot on which to draw

Notes

Uses the *title*, *cmap* and *bounds* properties of the *ImageParameter* class.

property enum_id
Returns the Parameter ID as an enumeration.

property parents
Parent *Parameter*s of this *Parameter*s instance.

set_bounds_lookup(*bounds_enum_id: enum.Enum*)
Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

property title
The image title used by the display function.

property type_info
The type of this *Parameter*.

property value
The value of this *Parameter*.

1.3.3 Parameter

class imgreg.util.params.**Parameter**(*enum_id: enum.Enum, value_definition: Union[Type[Any], Callable[[Dict[enum.Enum, imgreg.util.params.Parameter]], Any]], parent_parameters: Optional[Set[*imgreg.util.params.Parameter*]] = None*)
Bases: object

Base *Parameter* class, aware of related *Parameter*s and lazy evaluating.

Methods

__init__(enum_id, value_definition[, ...])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

add_child(*child:* imgreg.util.params.Parameter) → None
Assign a Parameter a child relation

add_descendant(*descendant:* imgreg.util.params.Parameter) → None
Assign a Parameter a child relation

property children
*Parameter*s with a child relation to this *Parameter*s instance.

clear(*clear_descendants=True*) → None
Clear the value stored in this and dependent *Parameter*s.

property const
Flag if the value can be overwritten once initialized.

property descendants
*Parameter*s with a descendant relation to this *Parameter*s instance.

property enum_id
Returns the Parameter ID as an enumeration.

property parents
Parent *Parameter*s of this *Parameter*s instance.

property type_info
The type of this *Parameter*.

property value
The value of this *Parameter*.

Exceptions

ParameterError
ParameterRegisterError

1.3.4 ParameterError

exception imgreg.util.params.**ParameterError**

1.3.5 ParameterRegisterError

exception imgreg.util.params.**ParameterRegisterError**

1.4 solver

Solvermodel primitives.

Author: Fabian A. Preiss

Functions

dependency_graph(parameters[, invert])	Construct a *DAGraph* dependency graph given a *Parameter* set.
dot_shape_func(parameter)	Generate the shape argument for a dot graph depending on the given Parameter
vertex_parent_dict_to_dot(vertex_parent_dict)	Convert a directed graph to a dot graph.

1.4.1 dependency_graph

imgreg.util.solver.**dependency_graph**(*parameters: Set[*imgreg.util.params.Parameter*]*, *invert=False*) → *imgreg.util.graph.DAGraph*

Construct a *DAGraph* dependency graph given a *Parameter* set.

1.4.2 dot_shape_func

imgreg.util.solver.**dot_shape_func**(*parameter:* imgreg.util.params.Parameter) → Dict[str, str]

Generate the shape argument for a dot graph depending on the given Parameter

Parameters

parameter [Parameter] The parameter of a node

1.4.3 vertex_parent_dict_to_dot

imgreg.util.solver.**vertex_parent_dict_to_dot**(*vertex_parent_dict: Dict[*imgreg.util.params.Parameter, *Set[*imgreg.util.params.Parameter*]]*, *node_args_func: Optional[Callable[[*imgreg.util.params.Parameter*], Dict[str, str]]] = None*, *invert=False*) → graphviz.dot.Digraph

Convert a directed graph to a dot graph.

Parameters

vertex_parent_dict [dict[Hashable, set[Hashable]]] A dictionary representation of the directed graph

node_args_func [Callable[[Hashable], dict[str, str]]] A function handle to generate keyword arguments for the node of the dot graph depending on the current vertex

Classes

Solver(parameters)	Interface for a Solvermodel constructed from a set of parameters.

1.4.4 Solver

class imgreg.util.solver.**Solver**(*parameters: Optional[Set[*imgreg.util.params.Parameter*]] = None*)

Bases: object

Interface for a Solvermodel constructed from a set of parameters.

Constructs a dependency graph for the *Parameter*s of the model and allows for lazy evaluation of the *Parameter*s.

Notes

The state of this class is cached, when a parameter is changed on which later states depend, the properties ascendant to said parameter are removed from the cache. *Solver._get_dep_graph()* allows access to the internal dependency graph representation.

Methods

__init__([parameters])	Initialize self.
display(param_list[, title])	Fancy plot functionality for registered ImageParameters.
dot_graph([node_args_func])	Return a dot graph representation of the solver model.

display(*param_list: Sequence[Union[enum.Enum,* imgreg.util.params.ImageParameter*]], title: Optional[str] = None*) → None

Fancy plot functionality for registered ImageParameters.

Parameters

plotlist [sequence] sequence, to access the ImageParameters registered in solver

title [str] str, contains title of overall plot

dot_graph(*node_args_func: Callable[[imgreg.util.params.Parameter], Dict[str, str]] = <function dot_shape_func>*) → graphviz.dot.Digraph

Return a dot graph representation of the solver model.

Exceptions

SolverError	

1.4.5 SolverError

exception imgreg.util.solver.**SolverError**

1.5 helpers

Collection of useful helper functions in specific usecases

Author: Fabian A. Preiss

Functions

image_file_gen(dview[, step])	Generate the images from a given DirectoryView as numpy arrays
image_save_back_tf(rot_tr_arr, fnames, ...)	Creates backtransformed images
rot_scale_tr_gen(solvers)	Sweep over solver instances and return the rotations, scales and translations
rot_tr_gen(solvers)	Sweep over solver instances and return *tr_x tr_y tr_err rot rot_err NormRel_L2*
solver_gen(dview, solver[, step])	Sweep the MOD_IMG parameters of a solver from a DirectoryView with a given stepsize.

1.5.1 image_file_gen

imgreg.util.helpers.**image_file_gen**(*dview: imgreg.util.io.DirectoryView*, *step: int = 1*) → Generator[numpy.ndarray, None, None]

Generate the images from a given DirectoryView as numpy arrays

Parameters

dview [DirectoryView] Generate images from top level folder

step [int] Stepsize in which files are taken

1.5.2 image_save_back_tf

imgreg.util.helpers.**image_save_back_tf**(*rot_tr_arr: numpy.ndarray*, *fnames: Sequence[str]*, *src_path: str*, *dest_path: str*)

Creates backtransformed images

For an external program that compares images **_A.** with **_B.** this prepares images such that:

```
test00001.jpg -> test00001_A.png
test00021.jpg -> TR -> test00001_B.png, test00021_A.png
```

(continues on next page)

(continued from previous page)

```
test00221.jpg -> TR -> test00201_B.png, test00221_A.png
test00241.jpg -> TR -> test00221_B.png
```

where TR denotes the backtransformation.

1.5.3 rot_scale_tr_gen

imgreg.util.helpers.**rot_scale_tr_gen**(*solvers: Iterable[*imgreg.util.solver.Solver*]*) → Generator[numpy.ndarray, None, None]

Sweep over solver instances and return the rotations, scales and translations

Parameters

solvers [Iterable] An iterable of solver instances

Returns

np.array Containing *tr_x tr_y tr_err rot rot_err scale scale_err NormRel_L2*

1.5.4 rot_tr_gen

imgreg.util.helpers.**rot_tr_gen**(*solvers: Iterable[*imgreg.util.solver.Solver*]*) → Generator[numpy.ndarray, None, None]

Sweep over solver instances and return *tr_x tr_y tr_err rot rot_err NormRel_L2*

Parameters

solvers [Iterable] An iterable of solver instances

1.5.5 solver_gen

imgreg.util.helpers.**solver_gen**(*dview:* imgreg.util.io.DirectoryView, *solver:* imgreg.util.solver.Solver, *step: int = 1*) → Generator[*imgreg.util.solver.Solver*, None, None]

Sweep the MOD_IMG parameters of a solver from a DirectoryView with a given stepsize.

Parameters

dview [DirectoryView] Generate images from top level folder

solver [Solver] Instance of a solver that

step [int] Stepsize in which files are taken

1.6 io

Directory processing utilities

Author: Fabian A. Preiss

Functions

fnmatch_filter(ls_files[, pattern])	Return the subset of strings that match given patterns.

1.6.1 fnmatch_filter

imgreg.util.io.**fnmatch_filter**(*ls_files: Set[str]*, *pattern: Union[str, List[str]] = '*'*) → Set[str]
Return the subset of strings that match given patterns.

Classes

DirectoryView([inputdir, file_pattern])	

1.6.2 DirectoryView

class imgreg.util.io.**DirectoryView**(*inputdir='.'*, *file_pattern='*'*)
Bases: object

Methods

__init__([inputdir, file_pattern])	Initialize self.
file_path_generator([step])	

Exceptions

DirectoryViewError	

1.6.3 DirectoryViewError

exception imgreg.util.io.**DirectoryViewError**

CHAPTER

TWO

MODELS

logpolar	Image registration based on the log-polar transform.
radon	Image registration based on the radon transform.
angleselect	Image selection by angle matching.
validator	Image validation properties.

2.1 logpolar

Image registration based on the log-polar transform.

Author: Fabian A. Preiss

Modules

imgreg.models.logpolar.enums	Enum of the logpolar parameters.
imgreg.models.logpolar.params	Module implementing the parameter classes for the LogPolarSolver Model
imgreg.models.logpolar.solver	The log-polar transform based image registration solver.

2.1.1 enums

Enum of the logpolar parameters.

Author: Fabian A. Preiss

Classes

LogPolParams(value)	An enumeration.

LogPolParams

class imgreg.models.logpolar.enums.**LogPolParams**(*value*)
An enumeration.

Attributes

BOUNDS	Boundaries for image slicing.
FOURIER_MOD_IMG	Fourier transformed and filtered modified image.
FOURIER_REF_IMG	Fourier transformed and filtered reference image.
GAUSS_DIFF	Lower and upper kernel size of the DoG filter.
GAUSS_DIFF_MOD_IMG	Gaussian difference filtered and windowed modified input image.
GAUSS_DIFF_REF_IMG	Gaussian difference filtered and windowed reference input image.
MOD_IMG	Modified image.
RECOVERED_ROTATION	Recovered rotation angle and error in degrees.
RECOVERED_ROTATION_SCALE_PHASE	Recovered rotation angle in degrees and scaling factor including errors.
RECOVERED_ROT_SCALE_IMG	Rotation and scaling recovered image.
RECOVERED_ROT_SCALE_TR_IMG	Rotation, scaling and translation recovered image.
RECOVERED_SCALE	Recovered scaling factor and error in degrees.
RECOVERED_TRANSLATION	Recovered x,y translation vector and error.
REF_IMG	Reference image.
UPSAMPLING	Upsampling factor.
WARPED_FOURIER_MOD_IMG	Log-polar transformed of the fourier transformed modified image.
WARPED_FOURIER_REF_IMG	Log-polar transformed of the fourier transformed reference image.
WINDOW_RADIUS_EXP	Window radius exponent, larger wrexp => faster computation.
WINDOW_TYPE	Window type for FFT filter.
WINDOW_WEIGHT	Weighting factor scaling beween windowed image and image, range of [0,1].

BOUNDS = 'BOUNDS'
Boundaries for image slicing.

FOURIER_MOD_IMG = 'FOURIER_MOD_IMG'
Fourier transformed and filtered modified image.

FOURIER_REF_IMG = 'FOURIER_REF_IMG'
Fourier transformed and filtered reference image.

GAUSS_DIFF = 'GAUSS_DIFF'
Lower and upper kernel size of the DoG filter.

GAUSS_DIFF_MOD_IMG = 'GAUSS_DIFF_MOD_IMG'
Gaussian difference filtered and windowed modified input image.

GAUSS_DIFF_REF_IMG = 'GAUSS_DIFF_REF_IMG'
Gaussian difference filtered and windowed reference input image.

MOD_IMG = 'MOD_IMG'

Modified image.

RECOVERED_ROTATION = 'RECOVERED_ROTATION'
Recovered rotation angle and error in degrees.

RECOVERED_ROTATION_SCALE_PHASE = 'RECOVERED_ROTATION_SCALE_PHASE'
Recovered rotation angle in degrees and scaling factor including errors.

RECOVERED_ROT_SCALE_IMG = 'RECOVERED_ROT_SCALE_IMG'
Rotation and scaling recovered image.

RECOVERED_ROT_SCALE_TR_IMG = 'RECOVERED_ROT_SCALE_TR_IMG'
Rotation, scaling and translation recovered image.

RECOVERED_SCALE = 'RECOVERED_SCALE'
Recovered scaling factor and error in degrees.

RECOVERED_TRANSLATION = 'RECOVERED_TRANSLATION'
Recovered x,y translation vector and error.

REF_IMG = 'REF_IMG'
Reference image.

UPSAMPLING = 'UPSAMPLING'
Upsampling factor.

WARPED_FOURIER_MOD_IMG = 'WARPED_FOURIER_MOD_IMG'
Log-polar transformed of the fourier transformed modified image.

WARPED_FOURIER_REF_IMG = 'WARPED_FOURIER_REF_IMG'
Log-polar transformed of the fourier transformed reference image.

WINDOW_RADIUS_EXP = 'WINDOW_RADIUS_EXP'
Window radius exponent, larger wrexp => faster computation.

WINDOW_TYPE = 'WINDOW_TYPE'
Window type for FFT filter.

WINDOW_WEIGHT = 'WINDOW_WEIGHT'
Weighting factor scaling beween windowed image and image, range of [0,1].

2.1.2 params

Module implementing the parameter classes for the LogPolarSolver Model

Author: Fabian A. Preiss

Classes

BoundsParam(parent_parameters, bounds, int, ...)	Boundaries for image slicing.
FourierModParam(parent_parameters[, ...])	Fourier transformed and filtered modified image.
FourierRefParam(parent_parameters[, ...])	Fourier transformed and filtered reference image.
GaussDiffParam(gauss_diff, float])	Lower and upper kernel size of the DoG filter.
GaussDiffWindowModParam(parent_parameters[, ...])	Gaussian difference filtered and windowed modified input image.
GaussDiffWindowRefParam(parent_parameters[, ...])	Gaussian difference filtered and windowed reference input image.

continues on next page

Table 5 – continued from previous page

ModImageParam(image)	Modified image.
RecoveredRotScaleParam(parent_parameters[, ...])	Rotation and scaling recovered image.
RecoveredRotScaleTr(parent_parameters[, ...])	Rotation, scaling and translation recovered image.
RecoveredRotationParam(parent_parameters[, ...])	Recovered rotation angle and error in degrees.
RecoveredRotationScalePhaseParam(...[, ...])	Recovered rotation angle in degrees and scaling factor including errors.
RecoveredScaleParam(parent_parameters[, ...])	Recovered scaling factor and error in degrees.
RecoveredTranslationParam(parent_parameters)	Recovered x,y translation vector and error.
RefImageParam(image)	Reference image.
UpsamplingParam(upsampling)	Upsampling factor.
WarpedFourierModParam(parent_parameters[, ...])	Log-polar transformed of the fourier transformed modified image.
WarpedFourierRefParam(parent_parameters[, ...])	Log-polar transformed of the fourier transformed reference image.
WindowRadiusExpParam(w_r_exp)	Window radius exponent, larger wrexp => faster computation.
WindowTypeParam(window_type)	Window type for FFT filter, see scipy.signal.windows.get_window for possible choices.
WindowWeightParam(window_weight)	Weighting factor scaling beween windowed image and image, range of [0,1].

BoundsParam

class imgreg.models.logpolar.params.**BoundsParam**(*parent_parameters*, *bounds: Optional[Tuple[int, int, int, int]] = None*)

Boundaries for image slicing.

Attributes

value [tuple[int, int, int, int]] The value of this *Parameter*.

Methods

__init__(parent_parameters[, bounds])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

FourierModParam

class imgreg.models.logpolar.params.**FourierModParam**(*parent_parameters*, *fourier_mod_img=None*)

Fourier transformed and filtered modified image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, fourier_mod_img])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.

continues on next page

Table 9 – continued from previous page

value	The value of this *Parameter*.

FourierRefParam

class imgreg.models.logpolar.params.**FourierRefParam**(*parent_parameters*, *fourier_ref_img=None*)

Fourier transformed and filtered reference image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, fourier_ref_img])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

GaussDiffParam

class imgreg.models.logpolar.params.**GaussDiffParam**(*gauss_diff: Tuple[float, float]*)

Lower and upper kernel size of the DoG filter.

Notes

Typical ratios for image enhancement are in the order of 1:4 and 1:5.[1] A ratio of 1.6 approximates the Laplacian of Gaussian filter.

References

Attributes

value [tuple[float, float]] The value of this *Parameter*.

Methods

__init__(gauss_diff)	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

GaussDiffWindowModParam

class imgreg.models.logpolar.params.**GaussDiffWindowModParam**(*parent_parameters*, *gauss_diff_mod_img=None*)

Gaussian difference filtered and windowed modified input image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

[1] Wikipedia, "Difference of Gaussians"

Methods

`__init__`(parent_parameters[, gauss_diff_mod_img])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
`display`([axsp])	Display the stored image using matplotlib.
`set_bounds_lookup`(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

`aspect`	The aspect ratio used by matplotlib.
`bounds`	Cropping boundaries for the display function.
`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`cmap`	The colormap used by matplotlib.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`title`	The image title used by the display function.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

GaussDiffWindowRefParam

class imgreg.models.logpolar.params.**GaussDiffWindowRefParam**(*parent_parameters*, *gauss_diff_ref_img=None*)

Gaussian difference filtered and windowed reference input image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, gauss_diff_ref_img])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
`display`([axsp])	Display the stored image using matplotlib.

continues on next page

Table 16 – continued from previous page

set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

ModImageParam

class imgreg.models.logpolar.params.**ModImageParam**(*image: Optional[numpy.ndarray] = None*)

Modified image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__([image])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RecoveredRotScaleParam

class imgreg.models.logpolar.params.**RecoveredRotScaleParam**(*parent_parameters*, *recovered_rot_scale_img=None*)

Rotation and scaling recovered image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, ...])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.

continues on next page

Table 21 – continued from previous page

enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RecoveredRotScaleTr

class imgreg.models.logpolar.params.**RecoveredRotScaleTr**(*parent_parameters*, *recovered_rot_scale_tr_img=None*)

Rotation, scaling and translation recovered image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, ...])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RecoveredRotationParam

class imgreg.models.logpolar.params.**RecoveredRotationParam**(*parent_parameters*, *recovered_rotation=None*)

Recovered rotation angle and error in degrees.

Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, recovered_rotation])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RecoveredRotationScalePhaseParam

class imgreg.models.logpolar.params.**RecoveredRotationScalePhaseParam**(*parent_parameters*, *recovered_rotation_scale_phase=No*

Recovered rotation angle in degrees and scaling factor including errors.

Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

Attributes

value [tuple[np.ndarray, np.ndarray, dict[str, Hashable]]] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, ...])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

RecoveredScaleParam

class imgreg.models.logpolar.params.**RecoveredScaleParam**(*parent_parameters*, *recovered_scale=None*)

Recovered scaling factor and error in degrees.

Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, recovered_scale])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

RecoveredTranslationParam

class imgreg.models.logpolar.params.**RecoveredTranslationParam**(*parent_parameters*, *recovered_translation=None*)

Recovered x,y translation vector and error.

Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, ...])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RefImageParam

class imgreg.models.logpolar.params.**RefImageParam**(*image: Optional[numpy.ndarray] = None*)

Reference image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__([image])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

UpsamplingParam

class imgreg.models.logpolar.params.**UpsamplingParam**(*upsampling: int*)

Upsampling factor. 1 => no upsampling, 20 => precision to 1/20 of a pixel.

Attributes

value [int] The value of this *Parameter*.

Methods

__init__(upsampling)	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

WarpedFourierModParam

class imgreg.models.logpolar.params.**WarpedFourierModParam**(*parent_parameters*, *warped_fourier_mod_img=None*)

Log-polar transformed of the fourier transformed modified image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, ...])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

WarpedFourierRefParam

class imgreg.models.logpolar.params.**WarpedFourierRefParam**(*parent_parameters*, *warped_fourier_ref_img=None*)

Log-polar transformed of the fourier transformed reference image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, ...])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.

continues on next page

Table 39 – continued from previous page

parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

WindowRadiusExpParam

class imgreg.models.logpolar.params.**WindowRadiusExpParam**(*w_r_exp: float*)

Window radius exponent, larger wrexp => faster computation.

Notes

If a value larger then 1 is used, this introduces a lowpass filter.

Attributes

value [float >= 1] The value of this *Parameter*.

Methods

__init__(w_r_exp)	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

WindowTypeParam

class imgreg.models.logpolar.params.**WindowTypeParam**(*window_type: str*)
Window type for FFT filter, see scipy.signal.windows.get_window for possible choices.

Attributes

value [str] The value of this *Parameter*.

Methods

__init__(window_type)	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

WindowWeightParam

class imgreg.models.logpolar.params.**WindowWeightParam**(*window_weight: float*)
Weighting factor scaling beween windowed image and image, range of [0,1].

Attributes

value [str] The value of this *Parameter*.

Methods

__init__(window_weight)	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

2.1.3 solver

The log-polar transform based image registration solver.

Author: Fabian A. Preiss

Classes

LogPolarSolver(ref_img, mod_img)	Implements an image registration model based on the log-polar transform.

LogPolarSolver

class imgreg.models.logpolar.solver.**LogPolarSolver**(*ref_img: Optional[numpy.ndarray] = None*, *mod_img: Optional[numpy.ndarray] = None*)

Implements an image registration model based on the log-polar transform.

The model tries to reconstruct the difference of scale, rotation and translation between two images.

Parameters

ref_img [numpy.ndarray] The original input image (one color channel only).

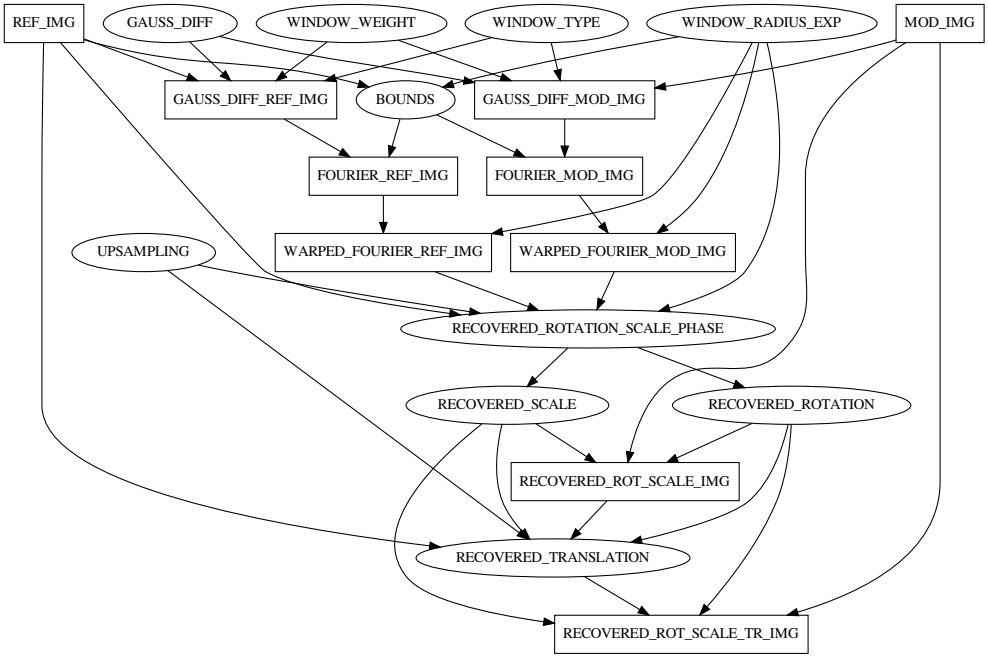
mod_img [numpy.ndarray] The modified input image (one color channel only).

Notes

Build based on the approach of the example code[1] from scikit-image. Alternative implementations using feature based detection algorithms are shown in[2].

The model implements the following dependency graph to construct it's *Parameter*s.

[1] Using Polar and Log-Polar Transformations for Registration
[2] ORB feature detector and binary descriptor



The *Parameter*s are documented in *params*.

References

Examples

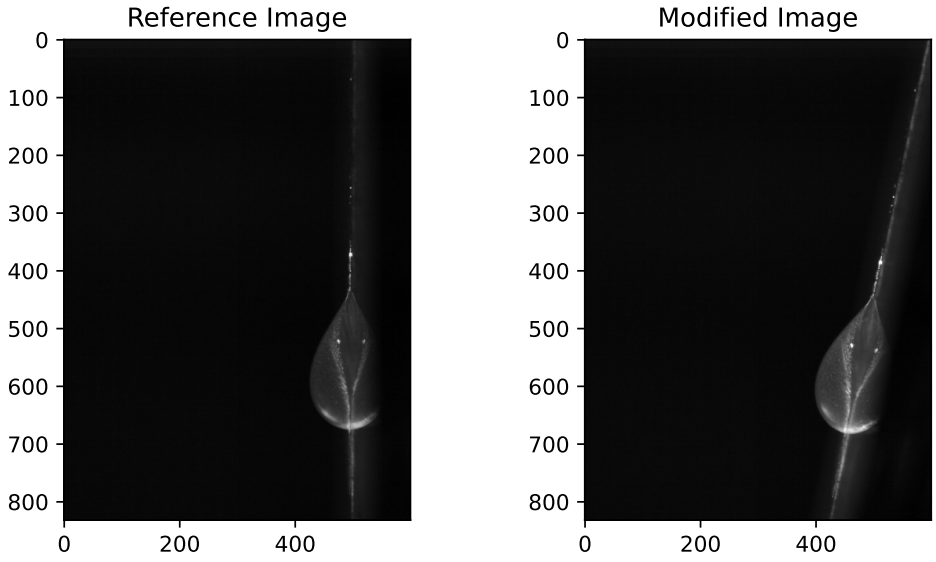
We can visualize the internal *ImageParameter*s of the model as follows:

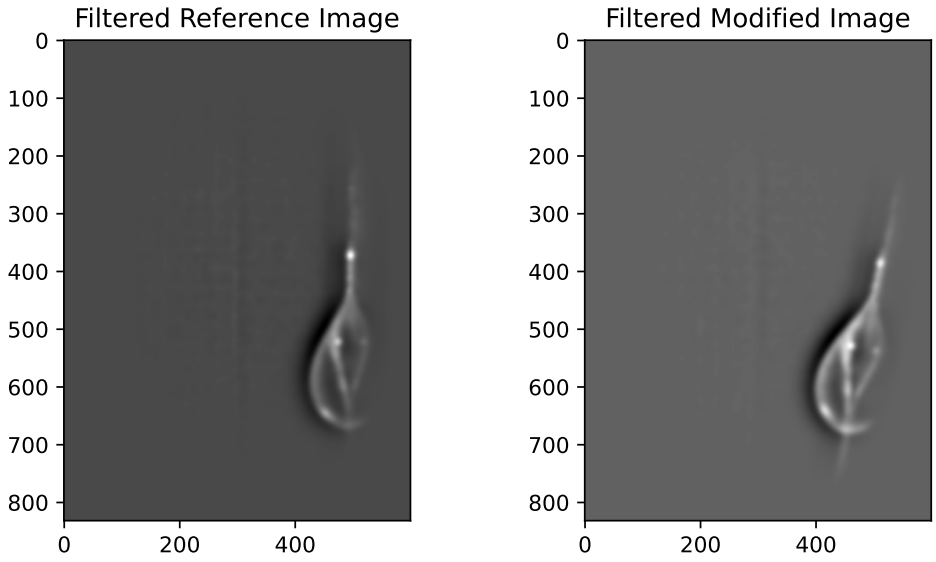
```
import numpy as np
import imgreg.data as data
from imgreg.models.logpolar import LogPolarSolver

ref_img = np.array(data.ref_img())
mod_img = np.array(data.mod_img())

# Create the model:
lps = LogPolarSolver(ref_img, mod_img)

# The ImageParameters of the model have matplotlib support via the display␣
↪function:
lps.display([lps.REF_IMG, lps.MOD_IMG])
lps.display([lps.GAUSS_DIFF_REF_IMG, lps.GAUSS_DIFF_MOD_IMG])
lps.display([lps.FOURIER_REF_IMG, lps.FOURIER_MOD_IMG])
lps.display([lps.WARPED_FOURIER_MOD_IMG, lps.WARPED_FOURIER_REF_IMG])
lps.display([lps.RECOVERED_ROT_SCALE_IMG, lps.REF_IMG])
lps.display([lps.RECOVERED_ROT_SCALE_TR_IMG, lps.REF_IMG])
```

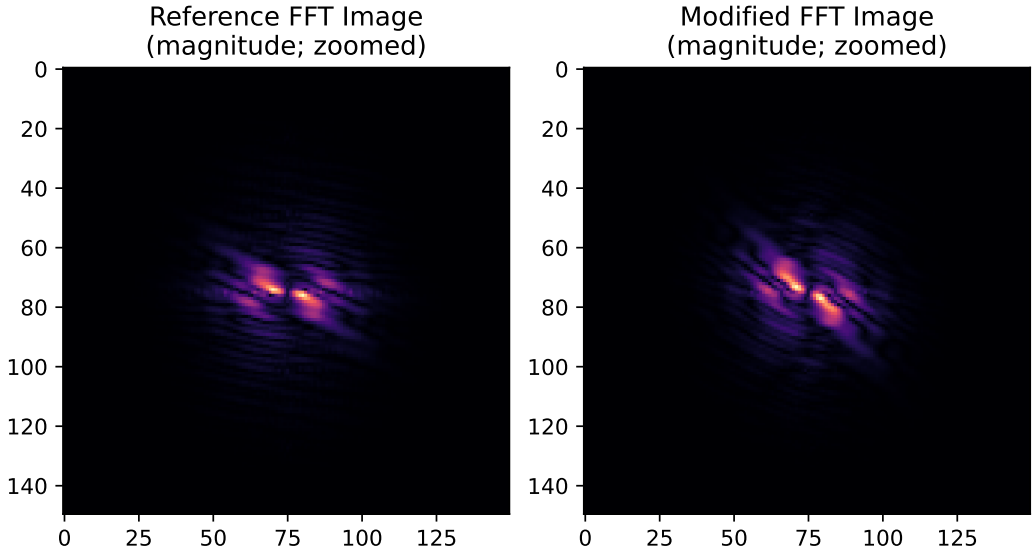
If we simply want to create a model and access the recovered values we first setup a model:

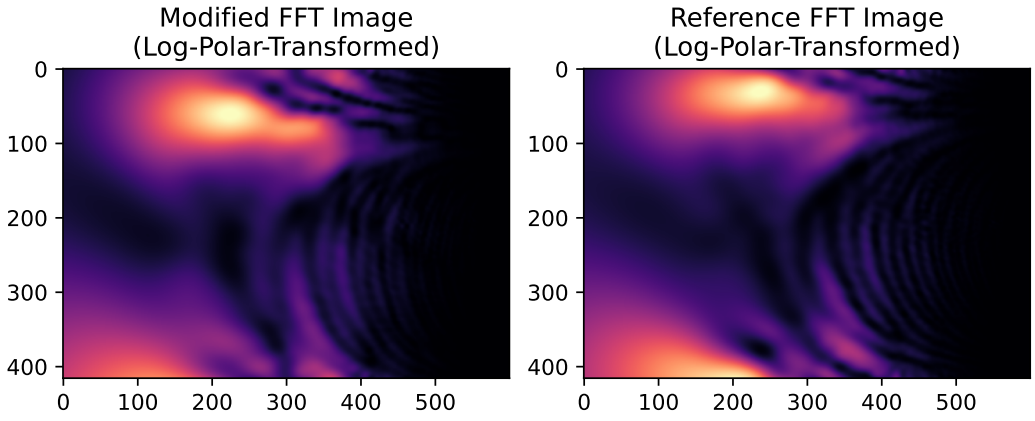
```
>>> import numpy as np
>>> import imgreg.data as data
```

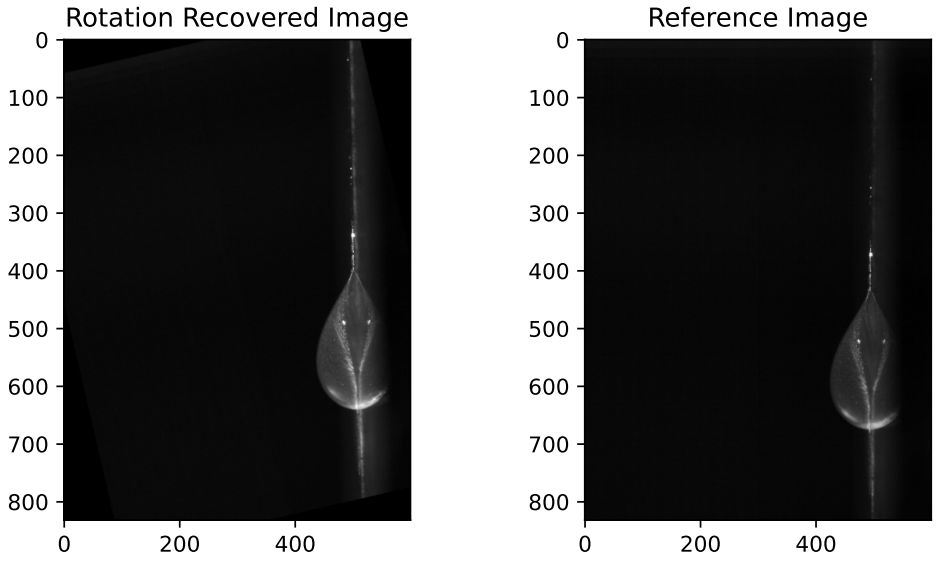
(continues on next page)

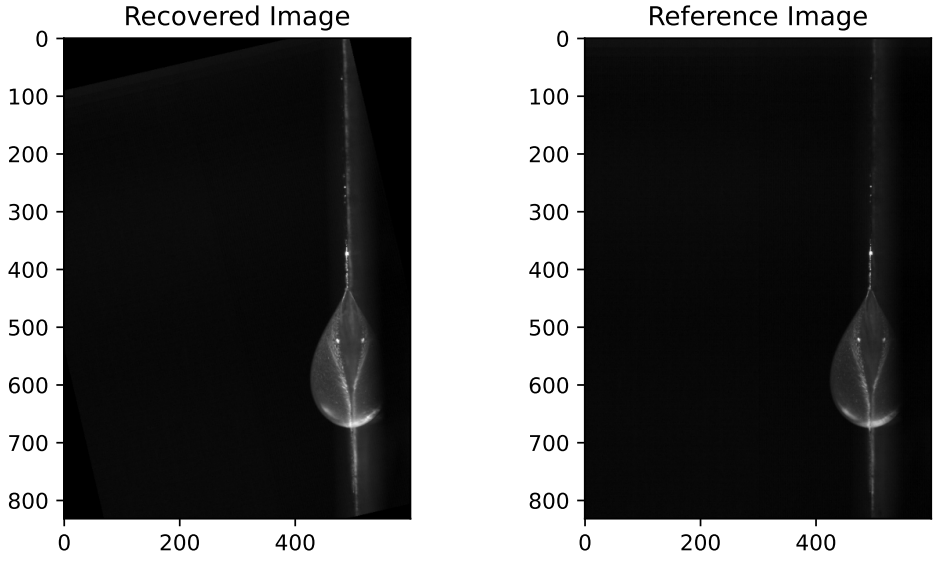

Reference Image
Modified Image
0
100
200
300
400
500
600
700
800
0
200
400


Filtered Reference Image
Filtered Modified Image
0
100
200
300
400
500
600
700
800
0
200
400


Reference FFT Image
(magnitude; zoomed)
Modified FFT Image
(magnitude; zoomed)
0
20
40
60
80
100
120
140
0
25
50
75
100
125


Modified FFT Image
(Log-Polar-Transformed)
Reference FFT Image
(Log-Polar-Transformed)
0
100
200
300
400
0
100
200
300
400
500


Rotation Recovered Image
Reference Image
0
100
200
300
400
500
600
700
800
0
200
400


Recovered Image
Reference Image
0
100
200
300
400
500
600
700
800
0
200
400

(continued from previous page)

```
>>> from imgreg.models.logpolar import LogPolarSolver, LogPolParams
>>> ref_img = np.array(data.ref_img())
>>> mod_img = np.array(data.mod_img())
>>> lps = LogPolarSolver(ref_img, mod_img)
```

Now the parameters of the model can now be accessed as follows:

```
>>> lps.RECOVERED_ROTATION.value
array([-13.06730769,   0.11259774])
```

```
>>> lps.RECOVERED_TRANSLATION.value
array([-17.98318062,  31.037803  ,   0.42407651])
```

Methods

`__init__`([ref_img, mod_img])	Initialize self.
`display`(param_list[, title])	Fancy plot functionality for registered ImageParameters.
`dot_graph`([node_args_func])	Return a dot graph representation of the solver model.

2.2 radon

Image registration based on the radon transform.

Author: Fabian A. Preiss

Modules

imgreg.models.radon.enums	Enum of the radon parameters.
imgreg.models.radon.params	Module implementing the parameter classes for the Radon Model
imgreg.models.radon.solver	The radon transform based image registration solver.

2.2.1 enums

Enum of the radon parameters.

Author: Fabian A. Preiss

Classes

RadonParams(value)	An enumeration.

RadonParams

class imgreg.models.radon.enums.**RadonParams**(*value*)
An enumeration.

Attributes

ANGLE_SELECT	A solver model for angle selection.
ANGULAR_PRECISION	Targeted angular precision in degrees.
EXPONENTIAL_FILTER_SIGNAL_NOISE	Signal to noise ratio when applying the min-max exponential filter.
MOD_IMG	Modified image.
MOD_ROTATION	Recovered rotation angle and error in degrees of the modified image.
RECOVERED_ROTATION	The recovered rotation angle and error between the modified and reference image.
RECOVERED_ROT_IMG	Rotation recovered image.
RECOVERED_ROT_TR_IMG	Rotation, and translation recovered image.
RECOVERED_TRANSLATION	Recovered x,y translation vector and error.
REF_IMG	Reference image.
REF_ROTATION	Recovered rotation angle and error in degrees of the reference image.
ROTATION_CANDIDATE	Candidate for the recovered rotation angle + error between the modified and reference image.
THETA	The initial search angles.
UPSAMPLING	Upsampling factor.

ANGLE_SELECT = 'ANGLE_SELECT'
A solver model for angle selection.

ANGULAR_PRECISION = 'ANGULAR_PRECISION'
Targeted angular precision in degrees.

EXPONENTIAL_FILTER_SIGNAL_NOISE = 'EXPONENTIAL_FILTER_SIGNAL_NOISE'
Signal to noise ratio when applying the min-max exponential filter.

MOD_IMG = 'MOD_IMG'
Modified image.

MOD_ROTATION = 'MOD_ROTATION'
Recovered rotation angle and error in degrees of the modified image.

RECOVERED_ROTATION = 'RECOVERED_ROTATION'
The recovered rotation angle and error between the modified and reference image.

RECOVERED_ROT_IMG = 'RECOVERED_ROT_IMG'
Rotation recovered image.

RECOVERED_ROT_TR_IMG = 'RECOVERED_ROT_TR_IMG'

Rotation, and translation recovered image.

RECOVERED_TRANSLATION = 'RECOVERED_TRANSLATION'
Recovered x,y translation vector and error.

REF_IMG = 'REF_IMG'
Reference image.

REF_ROTATION = 'REF_ROTATION'
Recovered rotation angle and error in degrees of the reference image.

ROTATION_CANDIDATE = 'ROTATION_CANDIDATE'
Candidate for the recovered rotation angle + error between the modified and reference image.

THETA = 'THETA'
The initial search angles.

UPSAMPLING = 'UPSAMPLING'
Upsampling factor.

2.2.2 params

Module implementing the parameter classes for the Radon Model

Author: Fabian A. Preiss

Classes

AngleSelectParam(parent_parameters, angle_select)	A solver model for angle selection.
AngularPrecisionParam(angular_precision)	Targeted angular precision in degrees.
ExponentialFilterSignalNoiseParam(...)	Signal to noise ratio when applying the min-max exponential filter.
ModImageParam(image)	Modified image.
ModRotationParam(parent_parameters, mod_rotation)	Recovered rotation angle and error in degrees of the modified image.
RecoveredRotParam(parent_parameters[, ...])	Rotation recovered image.
RecoveredRotTrParam(parent_parameters[, ...])	Rotation, and translation recovered image.
RecoveredRotationParam(parent_parameters, ...)	The recovered rotation angle and error between the modified and reference image.
RecoveredTranslationParam(parent_parameters)	Recovered x,y translation vector and error.
RefImageParam(image)	Reference image.
RefRotationParam(parent_parameters, ref_rotation)	Recovered rotation angle and error in degrees of the reference image.
RotationCandidateParam(parent_parameters, ...)	Candidate for the recovered rotation angle + error between the modified and reference image.
ThetaParam(theta)	The initial search angles.
UpsamplingParam(upsampling)	Upsampling factor.

AngleSelectParam

class imgreg.models.radon.params.**AngleSelectParam**(*parent_parameters*, *angle_select: Optional[imgreg.models.angleselect.solver.AngleSelect] = None*)

A solver model for angle selection.

Methods

`__init__`(parent_parameters[, angle_select])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

AngularPrecisionParam

class imgreg.models.radon.params.**AngularPrecisionParam**(*angular_precision: float*)

Targeted angular precision in degrees.

Notes

The targeted error is the lower estimate under ideal assumptions, in practise won't be achieavable.

Attributes

value [float] The value of this *Parameter*.

Methods

`__init__`(angular_precision)	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

ExponentialFilterSignalNoiseParam

class imgreg.models.radon.params.**ExponentialFilterSignalNoiseParam**(*exponential_filter_signal_noise: float*)

Signal to noise ratio when applying the min-max exponential filter.

Attributes

value [float] The value of this *Parameter*.

Methods

`__init__`(exponential_filter_signal_noise)	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.

continues on next page

Table 57 – continued from previous page

type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

ModImageParam

class imgreg.models.radon.params.**ModImageParam**(*image: Optional[numpy.ndarray] = None*)

Modified image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__([image])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

ModRotationParam

class imgreg.models.radon.params.**ModRotationParam**(*parent_parameters*, *mod_rotation: Optional[float] = None*)

Recovered rotation angle and error in degrees of the modified image.

Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, mod_rotation])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RecoveredRotParam

class imgreg.models.radon.params.**RecoveredRotParam**(*parent_parameters*, *recovered_rot_img=None*)

Rotation recovered image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, recovered_rot_img])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
`display`([axsp])	Display the stored image using matplotlib.
`set_bounds_lookup`(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

`aspect`	The aspect ratio used by matplotlib.
`bounds`	Cropping boundaries for the display function.
`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`cmap`	The colormap used by matplotlib.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`title`	The image title used by the display function.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

RecoveredRotTrParam

class imgreg.models.radon.params.**RecoveredRotTrParam**(*parent_parameters*, *recovered_rot_tr_img=None*)

Rotation, and translation recovered image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, ...])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
`display`([axsp])	Display the stored image using matplotlib.
`set_bounds_lookup`(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RecoveredRotationParam

class imgreg.models.radon.params.**RecoveredRotationParam**(*parent_parameters*, *angle_select: Optional[numpy.ndarray] = None*)

The recovered rotation angle and error between the modified and reference image.

Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, angle_select])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RecoveredTranslationParam

class imgreg.models.radon.params.**RecoveredTranslationParam**(*parent_parameters*, *recovered_translation=None*)

Recovered x,y translation vector and error.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, ...])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RefImageParam

class imgreg.models.radon.params.**RefImageParam**(*image: Optional[numpy.ndarray] = None*)

Reference image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__([image])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RefRotationParam

class imgreg.models.radon.params.**RefRotationParam**(*parent_parameters*, *ref_rotation: Optional[float] = None*)

Recovered rotation angle and error in degrees of the reference image.

Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

Attributes

value [float] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, ref_rotation])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

RotationCandidateParam

class imgreg.models.radon.params.**RotationCandidateParam**(*parent_parameters*, *rotation_candidate: Optional[numpy.ndarray] = None*)

Candidate for the recovered rotation angle + error between the modified and reference image.

Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, rotation_candidate])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

ThetaParam

class `imgreg.models.radon.params.`**ThetaParam**(*theta: numpy.ndarray*)

The initial search angles.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(theta)	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

UpsamplingParam

class imgreg.models.radon.params.**UpsamplingParam**(*upsampling: int*)

Upsampling factor. 1 => no upsampling, 20 => precision to 1/20 of a pixel.

Attributes

value [int] The value of this *Parameter*.

Methods

__init__(upsampling)	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

2.2.3 solver

The radon transform based image registration solver.

Author: Fabian A. Preiss

Classes

RadonSolver(ref_img, mod_img)	Implements an image registration model based on the radon transform.

RadonSolver

class imgreg.models.radon.solver.**RadonSolver**(*ref_img: Optional[numpy.ndarray] = None*, *mod_img: Optional[numpy.ndarray] = None*)

Implements an image registration model based on the radon transform.

The model tries to reconstruct the difference of rotation and translation between two images.

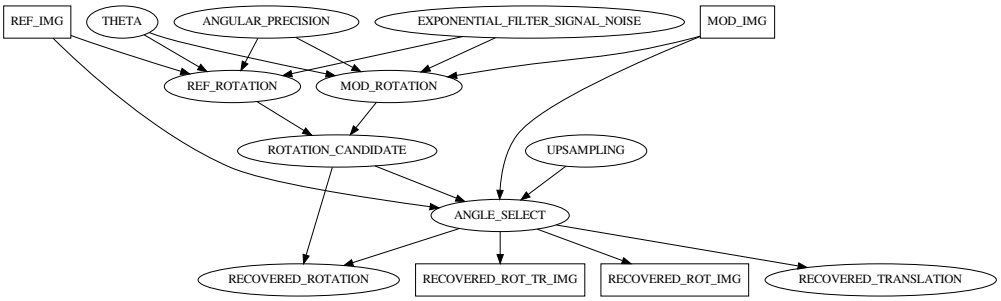
Parameters

ref_img [numpy.ndarray] The original input image (one color channel only).

mod_img [numpy.ndarray] The modified input image (one color channel only).

Notes

The model implements the following dependency graph to construct it's *Parameter*s.

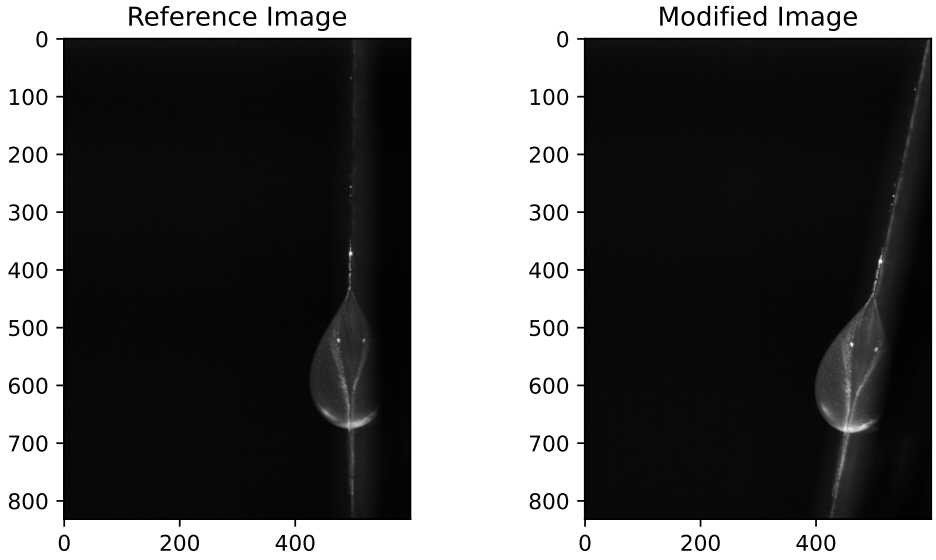


The *Parameter*s are documented in *params*.

Examples

We can visualize the internal *ImageParameter*s of the model as follows:

```
import numpy as np
import imgreg.data as data
from imgreg.models.radon import RadonSolver

ref_img = np.array(data.ref_img())
mod_img = np.array(data.mod_img())

# Create the model:
ras = RadonSolver(ref_img, mod_img)

# The ImageParameters of the model have matplotlib support via the display␣
↪function:
ras.display([ras.REF_IMG, ras.MOD_IMG])
ras.display([ras.RECOVERED_ROT_IMG, ras.REF_IMG])
ras.display([ras.RECOVERED_ROT_TR_IMG, ras.REF_IMG])
```

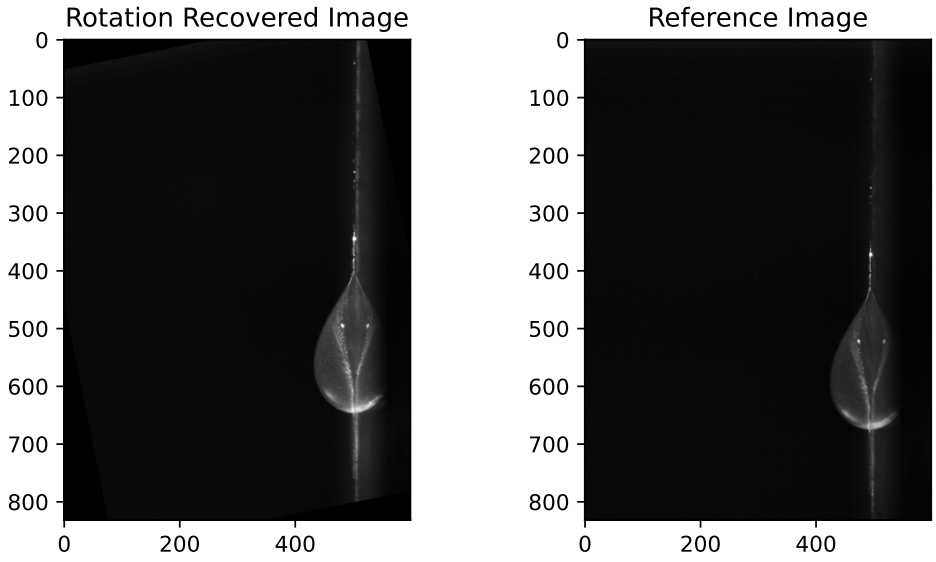


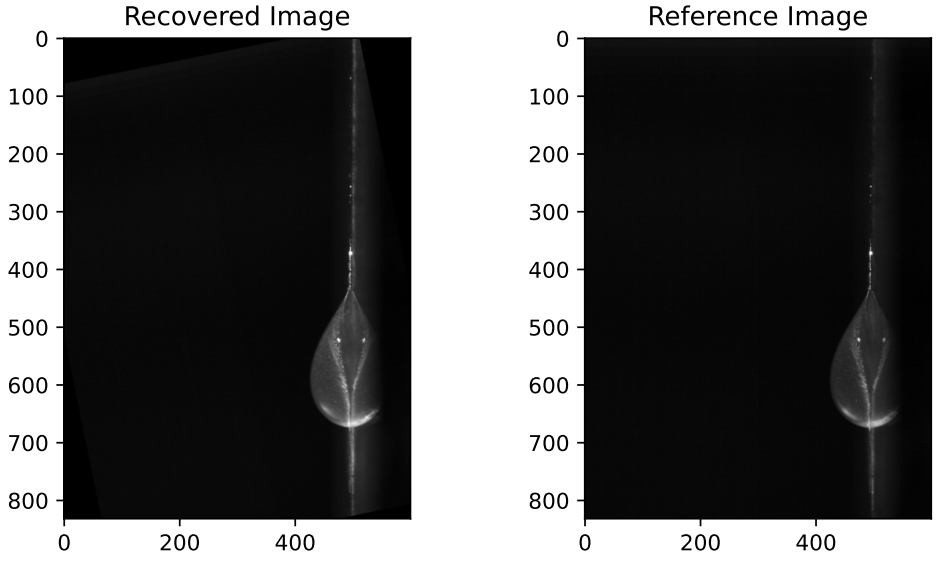
If we simply want to create a model and access the recovered values we first setup a model:

```
>>> import numpy as np
>>> import imgreg.data as data
>>> from imgreg.models.radon import RadonSolver
>>> ref_img = np.array(data.ref_img())
>>> mod_img = np.array(data.mod_img())
>>> ras = RadonSolver(ref_img, mod_img)
```

Now the parameters of the model can now be accessed as follows:

```
>>> ras.RECOVERED_ROTATION.value
array([-11.35802469,   0.28284271])
```


Rotation Recovered Image
Reference Image
0
100
200
300
400
500
600
700
800
0
200
400


Recovered Image
Reference Image
0
100
200
300
400
500
600
700
800
0
200
400

```
>>> ras.RECOVERED_TRANSLATION.value
array([-11.80822224,  25.77936941,   0.30466765])
```

Methods

`__init__`([ref_img, mod_img])	Initialize self.
`display`(param_list[, title])	Fancy plot functionality for registered ImageParameters.
`dot_graph`([node_args_func])	Return a dot graph representation of the solver model.

2.3 angleselect

Image selection by angle matching.

Author: Fabian A. Preiss

Modules

imgreg.models.angleselect.enums	Enum of the validator.
imgreg.models.angleselect.params	Module implementing the parameter classes for the angle selector Model
imgreg.models.angleselect.solver	Angle selection matching based on image matching.

2.3.1 enums

Enum of the validator.

Author: Fabian A. Preiss

Classes

AngleSelectParams(value)	An enumeration.

AngleSelectParams

class imgreg.models.angleselect.enums.**AngleSelectParams**(*value*)
An enumeration.

Attributes

ANGLE_A	Candidate Angle A.
ANGLE_B	Candidate Angle B.
IMG	Image.
RECOVERED_ROTATION	Recovered rotation angle and error between the modified and reference image.
RECOVERED_ROT_IMG	Rotation recovered image.
RECOVERED_ROT_TR_IMG	Rotation, and translation recovered image.
RECOVERED_TRANSLATION	Recovered x,y translation vector and error.
REF_IMG	Reference image.
ROT_A_IMG	Image Rotated by Angle A.
ROT_B_IMG	Image Rotated by Angle B.
ROT_TR_A_IMG	Rotation, and translation recovered image A.
ROT_TR_B_IMG	Rotation, and translation recovered image B.
SELECTOR	True, if A matches better than B
TRANSLATION_A	Recovered x,y translation vector and error for A.
TRANSLATION_B	Recovered x,y translation vector and error for B.
UPSAMPLING	Upsampling factor.

ANGLE_A = 'ANGLE_A'
Candidate Angle A.

ANGLE_B = 'ANGLE_B'
Candidate Angle B.

IMG = 'IMG'
Image.

RECOVERED_ROTATION = 'RECOVERED_ROTATION'
Recovered rotation angle and error between the modified and reference image.

RECOVERED_ROT_IMG = 'RECOVERED_ROT_SCALE_IMG'
Rotation recovered image.

RECOVERED_ROT_TR_IMG = 'RECOVERED_ROT_TR_IMG'
Rotation, and translation recovered image.

RECOVERED_TRANSLATION = 'RECOVERED_TRANSLATION'
Recovered x,y translation vector and error.

REF_IMG = 'REF_IMG'
Reference image.

ROT_A_IMG = 'ROT_A_IMG'
Image Rotated by Angle A.

ROT_B_IMG = 'ROT_B_IMG'
Image Rotated by Angle B.

ROT_TR_A_IMG = 'ROT_TR_A_IMG'
Rotation, and translation recovered image A.

ROT_TR_B_IMG = 'ROT_TR_B_IMG'
Rotation, and translation recovered image B.

SELECTOR = 'SELECTOR'
True, if A matches better than B

TRANSLATION_A = 'TRANSLATION_A'
Recovered x,y translation vector and error for A.

TRANSLATION_B = 'TRANSLATION_B'
Recovered x,y translation vector and error for B.

UPSAMPLING = 'UPSAMPLING'
Upsampling factor.

2.3.2 params

Module implementing the parameter classes for the angle selector Model

Author: Fabian A. Preiss

Classes

AngleAParam(angle_a)	Candidate Angle A.
AngleBParam(angle_b)	Candidate Angle B.
ImageParam(image)	Image.
RecoveredRotParam(parent_parameters[, ...])	Rotation recovered image.
RecoveredRotTrParam(parent_parameters[, ...])	Rotation, and translation recovered image.
RecoveredRotationParam(parent_parameters[, ...])	Recovered rotation angle and error between the modified and reference image.
RecoveredTranslationParam(parent_parameters)	Recovered x,y translation vector and error.
RefImageParam(image)	Reference image.
RotAParam(parent_parameters[, recovered_rot_img])	Image Rotated by Angle A.
RotBParam(parent_parameters[, recovered_rot_img])	Image Rotated by Angle B.
RotTrAParam(parent_parameters[, ...])	Rotation, and translation recovered image A.
RotTrBParam(parent_parameters[, ...])	Rotation, and translation recovered image B.
SelectorParam(parent_parameters[, selection])	True, if A matches better than B
TranslationAParam(parent_parameters[, ...])	Recovered x,y translation vector and error for A.
TranslationBParam(parent_parameters[, ...])	Recovered x,y translation vector and error for B.
UpsamplingParam(upsampling)	Upsampling factor.

AngleAParam

class imgreg.models.angleselect.params.**AngleAParam**(*angle_a: float*)
Candidate Angle A.

Attributes

value [float] The value of this *Parameter*.

Methods

`__init__`(angle_a)	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

AngleBParam

class imgreg.models.angleselect.params.**AngleBParam**(*angle_b: float*)

Candidate Angle B.

Attributes

value [float] The value of this *Parameter*.

Methods

`__init__`(angle_b)	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.

continues on next page

Table 89 – continued from previous page

value	The value of this *Parameter*.

ImageParam

class imgreg.models.angleselect.params.**ImageParam**(*image: Optional[numpy.ndarray] = None*)

Image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__([image])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RecoveredRotParam

class imgreg.models.angleselect.params.**RecoveredRotParam**(*parent_parameters*, *recovered_rot_img=None*)

Rotation recovered image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, recovered_rot_img])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RecoveredRotTrParam

class imgreg.models.angleselect.params.**RecoveredRotTrParam**(*parent_parameters*, *recovered_rot_tr_img=None*)

Rotation, and translation recovered image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, ...])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
`display`([axsp])	Display the stored image using matplotlib.
`set_bounds_lookup`(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

`aspect`	The aspect ratio used by matplotlib.
`bounds`	Cropping boundaries for the display function.
`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`cmap`	The colormap used by matplotlib.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`title`	The image title used by the display function.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

RecoveredRotationParam

class imgreg.models.angleselect.params.**RecoveredRotationParam**(*parent_parameters*, *recovered_rotation=None*)

Recovered rotation angle and error between the modified and reference image.

Notes

The errors are a lower estimate under ideal assumptions and can be much larger depending on the data.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, recovered_rotation])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

RecoveredTranslationParam

class imgreg.models.angleselect.params.**RecoveredTranslationParam**(*parent_parameters*, *recovered_translation=None*)

Recovered x,y translation vector and error.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, ...])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RefImageParam

class imgreg.models.angleselect.params.**RefImageParam**(*image: Optional[numpy.ndarray] = None*)

Reference image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__([image])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.

continues on next page

Table 101 – continued from previous page

value	The value of this *Parameter*.

RotAParam

class imgreg.models.angleselect.params.**RotAParam**(*parent_parameters*, *recovered_rot_img=None*)

Image Rotated by Angle A.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, recovered_rot_img])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RotBParam

class imgreg.models.angleselect.params.**RotBParam**(*parent_parameters*, *recovered_rot_img=None*)

Image Rotated by Angle B.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, recovered_rot_img])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

RotTrAParam

class imgreg.models.angleselect.params.**RotTrAParam**(*parent_parameters*, *recovered_rot_tr_img=None*)

Rotation, and translation recovered image A.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, ...])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
`display`([axsp])	Display the stored image using matplotlib.
`set_bounds_lookup`(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

`aspect`	The aspect ratio used by matplotlib.
`bounds`	Cropping boundaries for the display function.
`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`cmap`	The colormap used by matplotlib.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`title`	The image title used by the display function.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

RotTrBParam

class imgreg.models.angleselect.params.**RotTrBParam**(*parent_parameters*, *recovered_rot_tr_img=None*)

Rotation, and translation recovered image B.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, ...])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
`display`([axsp])	Display the stored image using matplotlib.
`set_bounds_lookup`(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

SelectorParam

class imgreg.models.angleselect.params.**SelectorParam**(*parent_parameters*, *selection=None*)

True, if A matches better than B

Attributes

value [bool] The value of this *Parameter*.

Methods

__init__(parent_parameters[, selection])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

TranslationAParam

class imgreg.models.angleselect.params.**TranslationAParam**(*parent_parameters*, *recovered_translation=None*)

Recovered x,y translation vector and error for A.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters[, ...])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

children	*Parameter*s with a child relation to this *Parameter*s instance.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

TranslationBParam

class imgreg.models.angleselect.params.**TranslationBParam**(*parent_parameters*, *recovered_translation=None*)

Recovered x,y translation vector and error for B.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`(parent_parameters[, ...])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

UpsamplingParam

class `imgreg.models.angleselect.params.`**`UpsamplingParam`**(*upsampling: int*)

Upsampling factor.

1 => no upsampling, 20 => precision to 1/20 of a pixel.

Attributes

`value` [int] The value of this *Parameter*.

Methods

`__init__`(upsampling)	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.

continues on next page

Table 117 – continued from previous page

parents	Parent *Parameter*s of this *Parameter*s instance.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

2.3.3 solver

Angle selection matching based on image matching.

Author: Fabian A. Preiss

Classes

AngleSelect(img, ref_img, angle_a, angle_b, ...)	Implements a model for angle selection

AngleSelect

class imgreg.models.angleselect.solver.**AngleSelect**(*img: numpy.ndarray*, *ref_img: numpy.ndarray*, *angle_a: float*, *angle_b: float*, *upsampling: int = 10*)

Implements a model for angle selection

Parameters

img [numpy.ndarray] The input image (one color channel only).

ref_img [numpy.ndarray] The reference image (one color channel only).

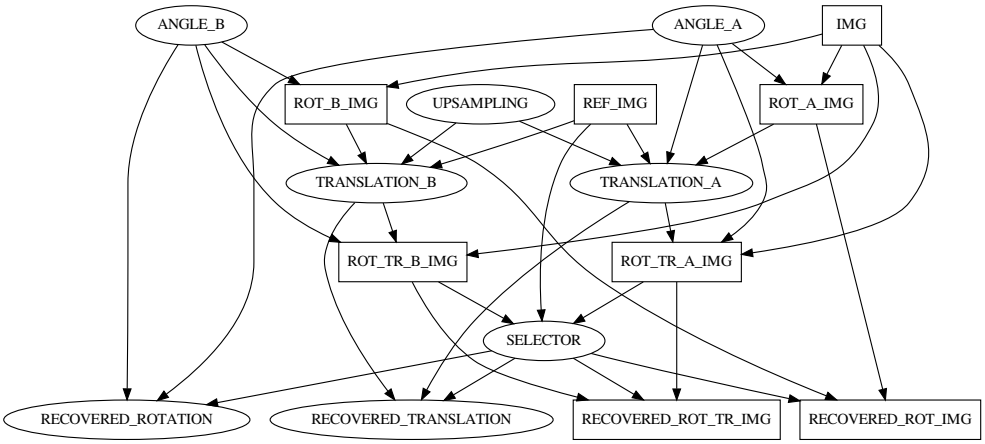
angle_a, angle_b [float] Candidate angles A and B in degrees

upsampling [int] Upsampling factor.

1 => no upsampling, 20 => precision to 1/20 of a pixel.

Notes

The model implements the following dependency graph to construct it's *Parameter*s.



Methods

`__init__`(img, ref_img, angle_a, angle_b[, ...])	Initialize self.
`display`(param_list[, title])	Fancy plot functionality for registered ImageParameters.
`dot_graph`([node_args_func])	Return a dot graph representation of the solver model.

2.4 validator

Image validation properties.

Author: Fabian A. Preiss

Modules

imgreg.models.validator.enums	Enum of the validator.
imgreg.models.validator.params	Module implementing the parameter classes for a validator.
imgreg.models.validator.solver	A validator for image similarity.

2.4.1 enums

Enum of the validator.

Author: Fabian A. Preiss

Classes

ValidatorParams(value)	An enumeration.

ValidatorParams

class imgreg.models.validator.enums.**ValidatorParams**(*value*)
An enumeration.

Attributes

ABSOLUTE_DIFFERENCE_IMG	Absolute value of the difference between the images.
IMG	Image.
NORM_REL_L2	Relative L2 Norm similarity measurement for the images.
REF_IMG	Reference image.
SQUARED_DIFFERENCE_IMG	Squared difference between the images.

ABSOLUTE_DIFFERENCE_IMG = 'ABSOLUTE_DIFFERENCE_IMG'
Absolute value of the difference between the images.

IMG = 'IMG'
Image.

NORM_REL_L2 = 'NORM_REL_L2'
Relative L2 Norm similarity measurement for the images.

REF_IMG = 'REF_IMG'
Reference image.

SQUARED_DIFFERENCE_IMG = 'SQUARED_DIFFERENCE_IMG'
Squared difference between the images.

2.4.2 params

Module implementing the parameter classes for a validator.

Author: Fabian A. Preiss

Classes

AbsoluteDifferenceParam(parent_parameters)	Absolute value of the difference between the images.
ImageParam(image)	Image.
NormRelL2Param(parent_parameters)	Relative L2 Norm similarity measurement for the images.
RefImageParam(image)	Reference image.
SquaredDifferenceParam(parent_parameters)	Squared difference between the images.

AbsoluteDifferenceParam

class imgreg.models.validator.params.**AbsoluteDifferenceParam**(*parent_parameters*)

Absolute value of the difference between the images.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters)	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

ImageParam

class imgreg.models.validator.params.**ImageParam**(*image: Optional[numpy.ndarray] = None*)

Image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__([image])	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

NormRelL2Param

class imgreg.models.validator.params.**NormRelL2Param**(*parent_parameters*)

Relative L2 Norm similarity measurement for the images.

Attributes

value [float] The value of this *Parameter*.

Methods

`__init__`(parent_parameters)	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.

Attributes

`children`	*Parameter*s with a child relation to this *Parameter*s instance.
`const`	Flag if the value can be overwritten once initialized.
`descendants`	*Parameter*s with a descendant relation to this *Parameter*s instance.
`enum_id`	Returns the Parameter ID as an enumeration.
`parents`	Parent *Parameter*s of this *Parameter*s instance.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

RefImageParam

class imgreg.models.validator.params.**RefImageParam**(*image: Optional[numpy.ndarray] = None*)

Reference image.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

`__init__`([image])	Initialize self.
`add_child`(child)	Assign a Parameter a child relation
`add_descendant`(descendant)	Assign a Parameter a child relation
`clear`([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
`display`([axsp])	Display the stored image using matplotlib.
`set_bounds_lookup`(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.
title	The image title used by the display function.
type_info	The type of this *Parameter*.
value	The value of this *Parameter*.

SquaredDifferenceParam

class imgreg.models.validator.params.**SquaredDifferenceParam**(*parent_parameters*)

Squared difference between the images.

Attributes

value [numpy.ndarray] The value of this *Parameter*.

Methods

__init__(parent_parameters)	Initialize self.
add_child(child)	Assign a Parameter a child relation
add_descendant(descendant)	Assign a Parameter a child relation
clear([clear_descendants])	Clear the value stored in this and dependent *Parameter*s.
display([axsp])	Display the stored image using matplotlib.
set_bounds_lookup(bounds_enum_id)	Reference a boundary Parameter using its *enum_id* as a lookup for cropping.

Attributes

aspect	The aspect ratio used by matplotlib.
bounds	Cropping boundaries for the display function.
children	*Parameter*s with a child relation to this *Parameter*s instance.
cmap	The colormap used by matplotlib.
const	Flag if the value can be overwritten once initialized.
descendants	*Parameter*s with a descendant relation to this *Parameter*s instance.
enum_id	Returns the Parameter ID as an enumeration.
parents	Parent *Parameter*s of this *Parameter*s instance.

continues on next page

Table 133 – continued from previous page

`title`	The image title used by the display function.
`type_info`	The type of this *Parameter*.
`value`	The value of this *Parameter*.

2.4.3 solver

A validator for image similarity.

Author: Fabian A. Preiss

Classes

Validator(img, ref_img)	Implements a validator model for image comparison.

Validator

class imgreg.models.validator.solver.**Validator**(*img: Optional[numpy.ndarray] = None*, *ref_img: Optional[numpy.ndarray] = None*)

Implements a validator model for image comparison.

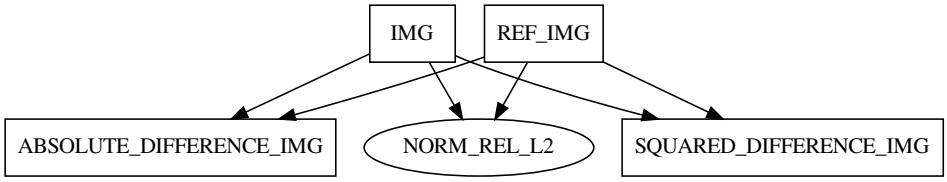
Parameters

img [numpy.ndarray] The input image (one color channel only).

ref_img [numpy.ndarray] The reference image (one color channel only).

Notes

The model implements the following dependency graph to construct it's *Parameter*s.

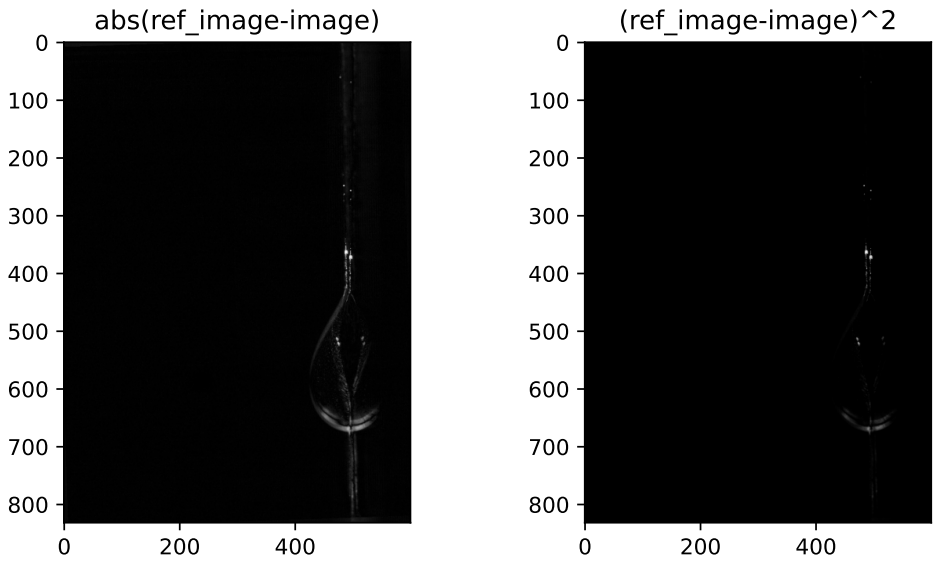


The *Parameter*s are documented in *params*.

Examples

We can visualize the internal *ImageParameter*s of the model as follows:

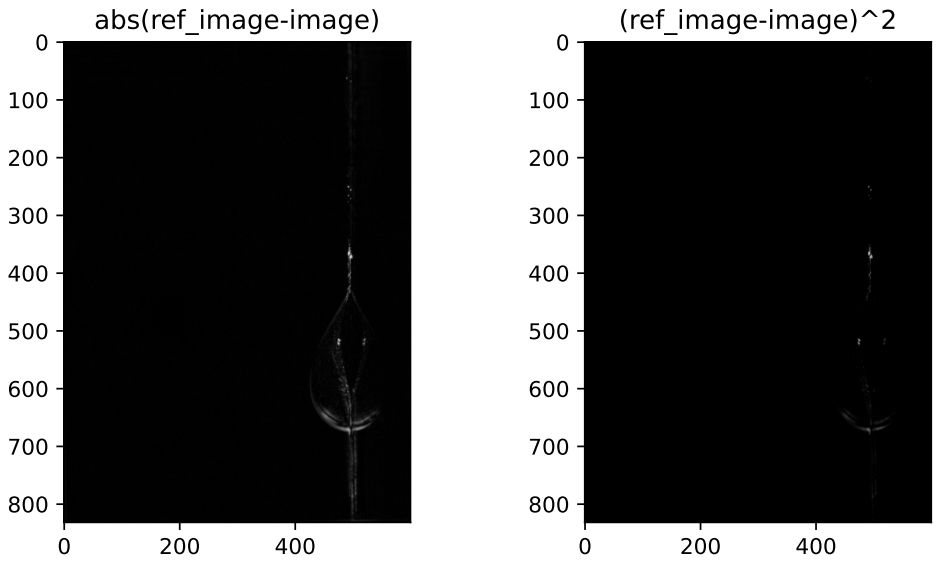
```
import numpy as np
import imgreg.data as data
from imgreg.models.validator import Validator
from imgreg.util.methods import ImageMethods

ref_img = np.array(data.ref_img())

# modify the image using an affine transformation
img = ImageMethods.compute_rts(ref_img, angle=2, translation=(6, 2))

# Create the model:
val = Validator(img, ref_img)

# The ImageParameters of the model have matplotlib support via the display␣
↪function:
val.display([val.ABSOLUTE_DIFFERENCE_IMG, val.SQUARED_DIFFERENCE_IMG])

# Increase the overlap to the reference image
val.IMG.value = ImageMethods.compute_rts(ref_img, angle=1, translation=(1, 2))

# Note how the difference images show less pronounced differences with increased␣
↪overlap
val.display([val.ABSOLUTE_DIFFERENCE_IMG, val.SQUARED_DIFFERENCE_IMG])
```



If we simply want to create a model and access the recovered values we first setup a model:

```
>>> import numpy as np
>>> import imgreg.data as data
>>> from imgreg.models.validator import Validator
>>> from imgreg.util.methods import ImageMethods
>>> ref_img = np.array(data.ref_img())
```

(continues on next page)



(continued from previous page)

```
>>> img = ImageMethods.compute_rts(ref_img, angle=2, translation=(6,2))
>>> val = Validator(img, ref_img)
```

Calculate the relative norm of difference between the images.

```
>>> val.NORM_REL_L2.value
0.4875079942792...
```

Note how this value approaches zero, as the image increase in their overlap:

```
>>> val.IMG.value = ImageMethods.compute_rts(ref_img, angle=1, translation=(1,2))
>>> val[ValidatorParams.NORM_REL_L2].value
0.3942652180108...
```

Methods

`__init__`([img, ref_img])	Initialize self.
`display`(param_list[, title])	Fancy plot functionality for registered ImageParameters.
`dot_graph`([node_args_func])	Return a dot graph representation of the solver model.

PYTHON MODULE INDEX

i

INDEX

A

B

C

D

E

F

G

I

L

M

N

P

R

S

T

U

V

W