RasterInLay

Release 0.1.2-rc

Jonas I. Liechti

Jan 05, 2021

CONTENTS:

CHAPTER

ONE

RASTERINLAY

1.1 Example

to do

1.2 rasterinlay

1.2.1 Submodules

1.2.1.1 rasterinlay.blocks

add_padding(*sblock: tuple*, *padding: int*)
Adding an padding to a block slice.

Parameters

- **sblock** – Slice of a block as returned by *.get_block*.
- **padding** – number of cells to add around the block slice.

breakdown_multiblocks(*multiblocks: Iterable[tuple]*, *block_shape: tuple*, *tolerance=1*) → list
Decomposes a collection of multi-blocks into a collection of blocks.

Parameters

- **multiblocks** – Collection of slices that specify multi-blocks.
- **block_shape** – Height and width (in number of pixles) of a block.
- **tolerance** – Accepted _increase_ in height and/or width of a block.

Returns Collection of normal blocks found in the collection of multi-blocks in *multiblocks*.

Return type list

breakdown_overloadeds(*raster*, *counts*, *overloadeds*, *block_shape*, *outvalue=0*, *tolerance=1*)
Decomposes a collection of overloaded blocks into classified blocks.

THIS IS NOT DONE!

Parameters

- **overloadeds** – Collection of slices that specify multi-blocks.
- **block_shape** – Height and width (in number of pixles) of a block.
- **tolerance** – Accepted _increase_ in height and/or width of a block.

Returns Collection of normal blocks found in the collection of multi-blocks in *multiblocks*.

Return type list

find_blocks(*raster: numpy.ndarray*, *block_shape: tuple*, *outvalue: Union[str, int, float] = 0*)
Identify all single-valued blocks within a raster.

Parameters

- **raster** – A 2D *.numpy.ndarray* in which blocks with an identical value should be found
- **block_shape** – Height and width (in number of pixles) of a block
- **outvalue** – The background value in *raster* that should be ignored in the same value counting.

Returns Collection of blocks found in raster. Blocks are specified with slices thus a single block can be passed as index to *raster*.

Return type list

get_block(*start: tuple*, *shape: tuple*, *offset: Optional[tuple] = None*) → tuple
Returns slice for accessing the block defined by *start* and *shape*.

Parameters

- **start** – Coordinates of the blocks top left corner
- **shape** – Height and width (in number of pixles) of the block
- **offset** – vertical and horizontal offset of *start*. The offset can be used if a block is given in another coordinate system, e.g. when a block is detected within another bock.
- **Returns** – Slice to retrieve the block.

get_counts(*raster: numpy.ndarray*, *outvalue: Union[str, int, float] = 0*)

get_superblocks(*raster: numpy.ndarray*, *counts: numpy.ndarray*, *outvalue: Union[str, int, float] = 0*)
→ Generator[tuple, None, None]

Parameters

- **raster** – A 2D `numpy.ndarray` in which blocks with an identical value should be found
- **counts** – 2D `numpy.ndarray` holding the counts of same valued diagonals.
- **outvalue** – The background value in *raster* that should be ignored in the same value counting.

Returns Collection of slices for all detected super-blocks. A super-block can be a block respecting a certain shape (determined when constructing *counts* - see *.get_counts* for details), a multi-block or an overloaded block. A multi-block contains only a single value but does not respect the limit shape. An overloaded block contains a single value in the top row and left column but other blocks within.

Return type Generator[tuple]

offset_block(*sblock: tuple*, *offset: tuple*)
Adding an offset to a block slice.

Parameters

- **sblock** – Slice of a block as returned by *.get_block*.
- **offset** – vertical and horizontal offset to add to the block slice.

separate_superblocks(*raster: numpy.ndarray*, *counts: numpy.ndarray*, *superblocks: Iterable[tuple]*, *block_shape: tuple*, *tolerance: int = 1*) → Tuple[list]

Parameters

- **raster** – A 2D array in which blocks with an identical value should be found
- **counts** – 2D *.numpy.ndarray* holding the counts of same valued diagonals
- **superblocks** – Collection of supeber-bock slices
- **block_shape** – Height and width (in number of pixles) of a block
- **tolerance** – Accepted _increase_ in height and/or width of a block.

Returns

blocks: list Collection of slices for all unique valued blocks respecting the *block_shape* (plus tolerance) constriaints

multi_blocks: list Collection of slices for all unique valued blocks that are over sized

overloaded_blocks: Collection of slices for all overloaded block. They contains a single value in the top row and left column but other blocks within.

Return type tuple

split_multiblock(*sblock: tuple*, *block_shape: tuple*, *tolerance: int = 1*) → list

Splits up a multi-block into a possible set blocks.

Parameters

- **sblock** – Slice of a block as returned by *.get_block*
- **block_shape** – Height and width (in number of pixles) of a block
- **tolerance** – Accepted _increase_ in height and/or width of a block.

Returns Collection of normal blocks that can compose the multi-block specified by the slices of *sblock*

Return type list

split_overloaded(*raster: numpy.ndarray*, *counts: numpy.ndarray*, *overloaded_block: tuple*, *block_shape: tuple*, *outvalue: Union[str, int, float] = 0*, *tolerance: int = 1*) → tuple

Splits an overloaded bock into super-blocks and classifies them.

Parameters

- **raster** – A 2D *.numpy.ndarray* in which blocks with an identical value should be found
- **counts** – 2D *.numpy.ndarray* holding the counts of same valued diagonals
- **overloaded_block** – Slices that determine an overloaded block in *raster*
- **block_shape** – Height and width (in number of pixles) of a block
- **outvalue** – The background value in *raster* that should be ignored in the same value counting
- **tolerance** – Accepted _increase_ in height and/or width of a block.

Returns

blocks: list Collection of slices for all unique valued blocks respecting the *block_shape* (plus tolerance) constriaints

multi_blocks: list Collection of slices for all unique valued blocks that are over sized

overloaded_blocks: Collection of slices for all overloaded block. They contains a single value in the top row and left column but other blocks within.

Return type tuple

1.2.1.2 rasterinlay.distribute

1.2.1.3 rasterinlay.helpers

1.2.1.4 rasterinlay.raredist

1.3 Installation

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX