# RadiPOP_API

Generated by Doxygen 1.9.2

# Chapter 1

# Namespace Index

## 1.1 Packages

Here are the packages with brief descriptions (if available):

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 3

# Class Index

## 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:

# Chapter 5

# Namespace Documentation

## 5.1 config Namespace Reference

### Classes

- class BaseConfig
- class DevelopmentConfig
- class ProductionConfig
- class TestingConfig

## 5.2 radipop_gui Namespace Reference

### Classes

- class RadiPopGUI

  *Bridge between the flask server/API and the RadiPOP segmenter:*

## 5.3 segmentation_utils Namespace Reference

### Functions

- def add_sobel_edges (mask, img)

  *Smooth edges Steps:*
- def draw_region_outlines (mask)

  *Color the mask light green.*
- def guess_bounds (regions_map, reference_map)

  *Guess the bounds/labels of the region based on reference region Guess the bounds/labels of the region based on reference region (generally neighboring slice).*
- def partition_at_threshold (img, thresh, square_size, min_size, title=None, show_plot=True)

  *After some smoothing, calculate new mask for img Steps:*
- def save_partition (mask, path)

### 5.3.1 Function Documentation

#### 5.3.1.1 add_sobel_edges()

```
def segmentation_utils.add_sobel_edges (
              mask,
              img )
```

Smooth edges Steps:

- Edge filter image using the Canny algorithm.
- euclidean distance transform

**Parameters**

| | |
|---|---|
| *mask* | mask corresponding to image |
| *img* | image corresponding to mask |

**Returns**

mask with smoothed edges

#### 5.3.1.2 draw_region_outlines()

```
def segmentation_utils.draw_region_outlines (
              mask )
```

Color the mask light green.

Color the edges of the mask darker green.

**Parameters**

| | |
|---|---|
| *mask* | mask for which to color the outlines |

**Returns**

mask with colored outlines

#### 5.3.1.3 guess_bounds()

```
def segmentation_utils.guess_bounds (
              regions_map,
              reference_map )
```

Guess the bounds/labels of the region based on reference region Guess the bounds/labels of the region based on reference region (generally neighboring slice).

**Parameters**

| | |
|---|---|
| *regions_map* | mask to guess labels for |
| *reference_map* | reference mask (already labelled) |

**Returns**

mask (labelled)

#### 5.3.1.4 partition_at_threshold()

```
def segmentation_utils.partition_at_threshold (
              img,
              thresh,
              square_size,
              min_size,
              title = None,
              show_plot = True )
```

After some smoothing, calculate new mask for img Steps:

- gaussian filter,
- remove small objects,
- greyscale morphological closing,
- euclidean distance transform

**Parameters**

| | |
|---|---|
| *img* | type numpy.ndarray |
| *thres* | Threshold value |
| *min_size* | Minimum size of an organ in the mask |
| *squaresize* | For greyscale morphological closing |
| *title* | Title of plot |
| *show_plot* | Show plot True/False |

**Returns**

New binary mask (same size as img)

#### 5.3.1.5 save_partition()

```
def segmentation_utils.save_partition (
              mask,
              path )
```

## 5.4 segmenter_flask_API Namespace Reference

### Functions

- def correctPartition ()

  *Reveives index to slice/mask + coordinates--> returns partion corrected mask as PNG to client.*
- def dcm2png ()

  *Receive Paths to dcm files, converts them to PNG.*
- def drawOnMask ()

  *Reveives index to slice/mask + x,y coordinates --> returns drawn on mask as PNG to client.*
- def extendThresholds ()

  *Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.*
- def getMask ()

  *Reveives index to slice/mask --> returns mask stored on flask server as PNG to client.*
- def highlightOrgan ()

  *Reveives index of slice + x,y coordinates --> returns highlighted mask as PNG to client.*
- def initialize ()

  *Receive Paths to ordered slices, caches slices.*
- def labelOrgan ()

  *Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.*
- def postPickleGetMask ()

  *Receives path to pickle file --> returns mask as PNG to client.*
- def saveMasks ()

  *Reveives path, saves all stored masks as pickle files to path --> returns output path.*
- def updateMask ()

  *Receives index of slice + slider values --> returns updated mask as PNG to client.*

### Variables

- app = Flask(__name__)
- int FLASK_PORT = 4041
- string FLAST_HOST = '0.0.0.0'
- host
- dictionary patients = {}

  *Dictionary which will hold for each patientID a RadiPopGUI object.*
- port

### 5.4.1 Function Documentation

#### 5.4.1.1 correctPartition()

```
def segmenter_flask_API.correctPartition ( )
```

Reveives index to slice/mask + coordinates--> returns partion corrected mask as PNG to client.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient |
| *index* | The index of the slice for which to mask should be updated |
| *coordinates* | array of coordinates of the form [x0,y0,x1,y1,...,xn,yn] |

**Returns**

JSON: {mask: byte stream} mask as transparent PNG as byte stream

Note: The coordinates array will be used to generate a line that cuts/divides the segmented organs.

Example handling of return image stream in js:

```
fetch(RadiPOP_states.FLASK_SERVER+"/labelOrgan", {
    method: 'POST',
    headers: { 'Content-Type': 'application/json'},
    body: JSON.stringify(data)
})
.then(function(response){ return response.json();})
.then(function(data){
bytestring = data["mask"];
img = bytestring.split('\'')[1]
```

#### 5.4.1.2 dcm2png()

```
def segmenter_flask_API.dcm2png ( )
```

Receive Paths to dcm files, converts them to PNG.

Included metadata IDs: "PatientID","PatientBirthDate","PatientName", "PatientAge","PatientSex","PatientName","↩ SliceThickness","StudyID","ContentDate"

**Parameters**

| | |
|---|---|
| *paths* | An array with the paths to the slices @parm low_clip: lowest pixel value |
| *high_clip* | highest pixel value |

**Returns**

JSON: {message: message, metadata: dictionary}

#### 5.4.1.3 drawOnMask()

```
def segmenter_flask_API.drawOnMask ( )
```

Reveives index to slice/mask + x,y coordinates --> returns drawn on mask as PNG to client.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient |
| *index* | The index of the slice for which to mask should be updated |
| *coordinates* | array of coordinates of the form [x0,y0,x1,y1,...,xn,yn] |

**Returns**

JSON: {mask: byte stream} mask as transparent PNG as byte stream

Note: The coordinates array will be used to draw a line on the mask.

Example handling of return image stream in js:

```
fetch(RadiPOP_states.FLASK_SERVER+"/labelOrgan", {
    method: 'POST',
    headers: { 'Content-Type': 'application/json'},
    body: JSON.stringify(data)
})
.then(function(response){ return response.json();})
.then(function(data){
bytestring = data["mask"];
img = bytestring.split('\'')[1]
```

#### 5.4.1.4 extendThresholds()

```
def segmenter_flask_API.extendThresholds ( )
```

Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient |
| *index* | The index of the slice for which to mask should be updated |
| *left* | Extend labeling up to index-label |
| *right* | Extend labeling up to index+label |

**Returns**

JSON: {left_most_idx: idx, right_most_idx: idx}

Note: The left_most_idx and right_most_idx correspond to the indices of the slices up to which the labeling has been extended. After the the function has finished use the function API's function /getMask to update the masks in your GUI. Example in js:

```
for (let index=parseInt(data["left_most_idx"]); index<parseInt(data["right_most_idx"])+1; index++) {
    $.post(FLASK_SERVER+"/getMask", {
        javascript_data: JSON.stringify({patienID: id, index: idx})
    })
}
```

#### 5.4.1.5 getMask()

```
def segmenter_flask_API.getMask ( )
```

Reveives index to slice/mask --> returns mask stored on flask server as PNG to client.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient |
| *index* | The index of the slice for which to mask should be updated |

**Returns**

JSON: {mask: byte stream} mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
fetch(RadiPOP_states.FLASK_SERVER+"/labelOrgan", {
    method: 'POST',
    headers: { 'Content-Type': 'application/json'},
    body: JSON.stringify(data)
})
.then(function(response){ return response.json();})
.then(function(data){
bytestring = data["mask"];
img = bytestring.split('\'')[1]
```

#### 5.4.1.6 highlightOrgan()

```
def segmenter_flask_API.highlightOrgan ( )
```

Reveives index of slice + x,y coordinates --> returns highlighted mask as PNG to client.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient |
| *index* | The index of the slice for which to mask should be updated |
| *x* | relative x coordinates (0<=x<=1) |
| *y* | relative y coordinates (0<=y<=1) |

**Returns**

JSON: {mask: byte stream} mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
fetch(RadiPOP_states.FLASK_SERVER+"/labelOrgan", {
    method: 'POST',
    headers: { 'Content-Type': 'application/json'},
    body: JSON.stringify(data)
})
.then(function(response){ return response.json();})
.then(function(data){
bytestring = data["mask"];
img = bytestring.split('\'')[1]
```

#### 5.4.1.7 initialize()

```
def segmenter_flask_API.initialize ( )
```

Receive Paths to ordered slices, caches slices.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient. Must be unique! |
| *paths* | An array with the paths to the slices |

**Returns**

JSON: {message: message}

Note: Paths to slices !!!MUST BE ORDERED!!! 0,1,..,n

#### 5.4.1.8 labelOrgan()

```
def segmenter_flask_API.labelOrgan ( )
```

Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient |
| *index* | The index of the slice for which to mask should be updated |
| *label* | Label of organ (1 for liver, 2 for spleen, 0 nothing, >2 other organ) |

**Returns**

JSON: {mask: byte stream} mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
fetch(RadiPOP_states.FLASK_SERVER+"/labelOrgan", {
    method: 'POST',
    headers: { 'Content-Type': 'application/json'},
    body: JSON.stringify(data)
})
.then(function(response){ return response.json();})
.then(function(data){
bytestring = data["mask"];
img = bytestring.split('\'')[1]
```

#### 5.4.1.9 postPickleGetMask()

```
def segmenter_flask_API.postPickleGetMask ( )
```

Receives path to pickle file --> returns mask as PNG to client.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient |
| *index* | The index of the slice the mask refers to |
| *path* | The path to the mask file |

**Returns**

JSON: {mask: byte stream} mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
fetch(RadiPOP_states.FLASK_SERVER+"/labelOrgan", {
    method: 'POST',
    headers: { 'Content-Type': 'application/json'},
    body: JSON.stringify(data)
})
.then(function(response){ return response.json();})
.then(function(data){
bytestring = data["mask"];
img = bytestring.split('\'')[1]
```

#### 5.4.1.10 saveMasks()

```
def segmenter_flask_API.saveMasks ( )
```

Reveives path, saves all stored masks as pickle files to path --> returns output path.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient |
| *index* | The index of the slice for which to mask should be updated |

**Returns**

JSON: {outdir: path} path/directory to which the pickle files were written

#### 5.4.1.11 updateMask()

```
def segmenter_flask_API.updateMask ( )
```

Receives index of slice + slider values --> returns updated mask as PNG to client.

**Parameters**

| | |
|---|---|
| *patientID* | The ID of the patient |
| *index* | The index of the slice for which to mask should be updated |
| *liver-intensity-slider* | Slider value for liver intesity |
| *bone-intensity-slider* | Slider value for bone intesity |
| *blood-vessel-intensity-slider* | Slider value for blood-vessel intesity |

**Returns**

JSON: {mask: byte stream} mask as transparent PNG as byte stream

Example handling of return image stream in js:

```
fetch(RadiPOP_states.FLASK_SERVER+"/labelOrgan", {
    method: 'POST',
    headers: { 'Content-Type': 'application/json'},
    body: JSON.stringify(data)
})
.then(function(response){ return response.json();})
.then(function(data){
bytestring = data["mask"];
img = bytestring.split('\'')[1]
```

### 5.4.2 Variable Documentation

#### 5.4.2.1 app

```
app = Flask(__name__)
```

#### 5.4.2.2 FLASK_PORT

```
int FLASK_PORT = 4041
```

#### 5.4.2.3 FLAST_HOST

```
FLAST_HOST = '0.0.0.0'
```

#### 5.4.2.4 host

```
host
```

#### 5.4.2.5 patients

```
dictionary patients = {}
```

Dictionary which will hold for each patientID a RadiPopGUI object.

Patients are added by the API's /initialize function

#### 5.4.2.6 port

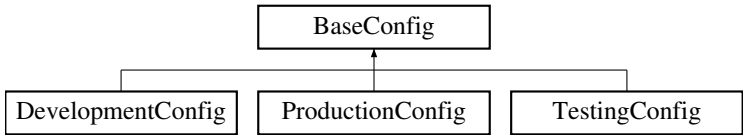
```
port
```

# Chapter 6

# Class Documentation

## 6.1 BaseConfig Class Reference

Inheritance diagram for BaseConfig:




### Static Public Attributes

- SECRET_KEY = os.getenv('SECRET_KEY', 'REPLACE ME')

### 6.1.1 Member Data Documentation

#### 6.1.1.1 SECRET_KEY

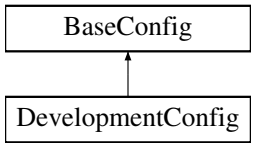
```
SECRET_KEY = os.getenv('SECRET_KEY', 'REPLACE ME')  [static]
```

The documentation for this class was generated from the following file:

- /Users/lorenz/Desktop/temp4/radipop/RadiPOP-standalone/web_app/utility/config.py

## 6.2 DevelopmentConfig Class Reference

Inheritance diagram for DevelopmentConfig:



### Static Public Attributes

- bool DEBUG = True

### 6.2.1 Member Data Documentation

#### 6.2.1.1 DEBUG

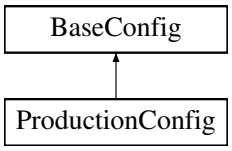
```
bool DEBUG = True  [static]
```

The documentation for this class was generated from the following file:

- /Users/lorenz/Desktop/temp4/radipop/RadiPOP-standalone/web_app/utility/config.py

## 6.3 ProductionConfig Class Reference

Inheritance diagram for ProductionConfig:




### Additional Inherited Members

The documentation for this class was generated from the following file:

- /Users/lorenz/Desktop/temp4/radipop/RadiPOP-standalone/web_app/utility/config.py

## 6.4 RadiPopGUI Class Reference

Bridge between the flask server/API and the RadiPOP segmenter:

## Public Member Functions

- def __init__ (self, patient_id)

  *Class constructor:*
- def extend_labels (self, cur_idx, left_extend, right_extend)

  *Extend labels from current slice to neighbouring slices.*
- def extract_metadata_from_dcm (dcm_file)

  *Read dicom image (.dcm) and extract metadata Extracted metadata IDs: "PatientID","PatientBirthDate","Patient↩ Name", "PatientAge","PatientSex","PatientName","SliceThickness","StudyID","ContentDate".*
- def highlightOrgan (self, slice_idx, x, y)

  *Highlights regions of the mask (organs) that were clicked on by user.*
- def labelMask (self, slice_idx, label)

  *Labels mask at given index at previously selected region.*
- def save_masks (self, path)

  *Saves masks as pickle file to given path.*
- def slice_dim (self)

  *Returns dimensions of slice images (x,y)*

## Static Public Member Functions

- def clip_dcm (dcm_file, clip_low=850, clip_high=1250)

  *Read dicom image (.dcm), clips it and returns it as a grey scale PNG.*
- def correct_partition (image)

  *Convert PNG mask to 1 channelled label mask.*
- def create_image_stream (img)

  *Returns base64 bytestream for given input image.*
- def draw_on_image (coordinates, img, correctionMode=False)

  *Draws a point or lines on given PNG image.*
- def find_organs (img, bones_thresh, blood_vessels_thresh, liver_thresh)

  *Uses three threshold values to find organs.*
- def np_label_array_to_png (mask_np_array, highlight=None)

  *Takes an numpy label array dim(n,m,1) and returns a RGBA pillow image dim(n,m,4)*
- def read_pickle_mask_to_np_label_array (path)

  *Opens mask pickle file and returns it as a np.array.*
- def readPNG (path)

  *Reads an image (e.g.*
- def update_mask_upon_slider_change (image, bone_intensity, blood_vessel_intensity, liver_intensity)

  *Sets threshold for liver intensity.*
- def writePillow2PNG (img, outfile)

## Public Attributes

- last_clicked_x

  *The x-coordinate of the region that was last selected/clicked on.*
- last_clicked_y

  *The y-coordinate of the region that was last selected/clicked on.*
- masks

  *Dictionary: key: mask index, value: mask numpy array dim(n,m,1)*
- pathToSlices

*List contaning the path to the png slice files*

- patient_id

  *ID of the patient.*
- selected_pixel_value_of_label_mask

  *The label of the region that was last selected/clicked on.*
- sliceCache

  *Dictionary: key: slice index, value: slice PNG.*

## Static Public Attributes

- int LIVER_LABEL = 1

  *Regions in mask with this label value are considered liver.*
- int SPLEEN_LABEL = 2

  *Regions in mask with this label value are considered spleen.*

### 6.4.1 Detailed Description

Bridge between the flask server/API and the RadiPOP segmenter:

Note:

- For each patient one object of this class should be instantiated.
- An object of this class contains all the slices and masks associated with the patient
- This class also contains static utility functions
- This class is the bridge between the flask server/API and the RadiPOP segmenter (segmentations_utils)

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 __init__()

```
def __init__ (
              self,
              patient_id )
```

Class constructor:

**Parameters**

| | |
|---|---|
| *patient↩ _id* | The ID of the patient. Must be unique! |

Note:

- For each patient one object of this class should be instantiated.
- An object of this class contains all the slices and masks associated with the patient

### 6.4.3 Member Function Documentation

#### 6.4.3.1 clip_dcm()

```
def clip_dcm (
              dcm_file,
              clip_low = 850,
              clip_high = 1250 )  [static]
```

Read dicom image (.dcm), clips it and returns it as a grey scale PNG.

```
    @param dcm_file Path to .dcm file
    @param clip_low lowest pixel value
    @param clip_high highest pixel value

    @return tuple(L (grey scale) Pillow Image, slice index)
```

#### 6.4.3.2 correct_partition()

```
def correct_partition (
              image )  [static]
```

Convert PNG mask to 1 channelled label mask.

**Parameters**

| | |
|---|---|
| *image* | Image (e.g.: RGBA PNG pillow) |

**Returns**

label mask

#### 6.4.3.3 create_image_stream()

```
def create_image_stream (
              img )  [static]
```

Returns base64 bytestream for given input image.

**Parameters**

| | |
|---|---|
| *img* | Image (e.g. Pillow PNG) |

**Returns**

img_base64 stream

#### 6.4.3.4 draw_on_image()

```
def draw_on_image (
              coordinates,
              img,
              correctionMode = False )  [static]
```

Draws a point or lines on given PNG image.

**Parameters**

| | |
|---|---|
| *coordinates* | List of the form [x0,y0,x1,y1,...,xn,yn] |
| *img* | Image (e.g.: RGBA PNG pillow) |
| *correctionMode* | True/False If true the drawn line acts as an eraser (dividing organs). If false a colored line is drawn on the image. DEFAULT: False |

The modifications are made directly on the provided image. No return value

#### 6.4.3.5 extend_labels()

```
def extend_labels (
              self,
              cur_idx,
              left_extend,
              right_extend )
```

Extend labels from current slice to neighbouring slices.

**Parameters**

| | |
|---|---|
| *cur_idx* | Index of reference slice |
| *left_extend* | Number of slices to extend the labeling to below reference slice |
| *right_extend* | Number of slices to extend the labeling to above reference slice |

**Returns**

(left_most_idx,right_most_idx) The index of the outermost slices to which the labeling was extended

Extends labels left and right from current slice How far the labels are extended is taken from left and right expansion bounds

#### 6.4.3.6 extract_metadata_from_dcm()

```
def extract_metadata_from_dcm (
              dcm_file )
```

Read dicom image (.dcm) and extract metadata Extracted metadata IDs: "PatientID","PatientBirthDate","Patient↩ Name", "PatientAge","PatientSex","PatientName","SliceThickness","StudyID","ContentDate".

**Parameters**

| | |
|---|---|
| *dcm_file* | Path to .dcm file |

**Returns**

dictionary with metadata information

#### 6.4.3.7 find_organs()

```
def find_organs (
              img,
              bones_thresh,
              blood_vessels_thresh,
              liver_thresh )  [static]
```

Uses three threshold values to find organs.

**Parameters**

| | |
|---|---|
| *image* | Image (e.g.: RGBA PNG pillow) |
| *bones_thresh* | bones threshold: [threshold, square_size , min_size] |
| *blood_vessels_thresh* | blood vessels threshold: [threshold, square_size , min_size] |
| *liver_thresh* | liver threshold: [threshold, square_size , min_size] |

**Returns**

New labelled mask (same size as slice)

```
The algorithm is:
    - After some smoothing, remove every pixel above bones threshold from the image.
    - After some smoothing, remove every pixel above blood vessel threshold.
    - Everything that then remains above liver threshold is called an organ.
    - Use contiguous area divisions to roughly split into organs.
```

#### 6.4.3.8 highlightOrgan()

```
def highlightOrgan (
              self,
              slice_idx,
              x,
              y )
```

Highlights regions of the mask (organs) that were clicked on by user.

**Parameters**

| | |
|---|---|
| *slice_idx* | Index of mask/slice to be highlighted |
| *x* | x-coordinate of slice (in pixels) |
| *y* | y-coordinate of slice (in pixels) |

**Returns**

Mask where the region specified by x and y is highlighted in brigther colors

#### 6.4.3.9 labelMask()

```
def labelMask (
              self,
              slice_idx,
              label )
```

Labels mask at given index at previously selected region.

**Parameters**

| | |
|---|---|
| *slice_idx* | Index of mask/slice to be labelled |
| *label* | Label to be assigned to previously selected region (either LIVER_LABEL or SPLEEN_LABEL) |

**Returns**

Mask with new label

Note: It is expected that the client has before highlighted an organ with the function self.highlightOrgan(). This determines the region/organ that will be labelled with label.

#### 6.4.3.10 np_label_array_to_png()

```
def np_label_array_to_png (
              mask_np_array,
              highlight = None )  [static]
```

Takes an numpy label array dim(n,m,1) and returns a RGBA pillow image dim(n,m,4)

**Parameters**

| | |
|---|---|
| *mask_np_array* | numpy label array dim(n,m,1) @highlight Highlight regions where highlight==label in brighter color |

**Returns**

RGBA pillow image dim(n,m,4)

Turns a labelled mask into a transparent (RGBA) PNG.

- Default liver color is red (LIVER_LABEL)
- Default spleen color is blue (SPLEEN_LABEL)
- Default for other regions is green

#### 6.4.3.11 read_pickle_mask_to_np_label_array()

```
def read_pickle_mask_to_np_label_array (
              path )  [static]
```

Opens mask pickle file and returns it as a np.array.

**Parameters**

| | |
|---|---|
| *path* | Path to pickle file |

**Returns**

numpy array of mask

#### 6.4.3.12 readPNG()

```
def readPNG (
              path )  [static]
```

Reads an image (e.g.

: PNG file) to numpy array

**Parameters**

| | |
|---|---|
| *path* | Path to image |

**Returns**

numpy array of image

#### 6.4.3.13 save_masks()

```
def save_masks (
              self,
              path )
```

Saves masks as pickle file to given path.

**Parameters**

| | |
|---|---|
| *path* | Path to which mask files should be written |

Masks are written as .p (pickle) files

#### 6.4.3.14 slice_dim()

```
def slice_dim (
              self )
```

Returns dimensions of slice images (x,y)

**Returns**

(x,y) Dimensions of slices

#### 6.4.3.15 update_mask_upon_slider_change()

```
def update_mask_upon_slider_change (
              image,
              bone_intensity,
              blood_vessel_intensity,
              liver_intensity )  [static]
```

Sets threshold for liver intensity.

**Parameters**

| | |
|---|---|
| *image* | Image (e.g.: RGBA PNG pillow) |
| *bones_thresh* | bones threshold |
| *blood_vessels_thresh* | blood vessels threshold |
| *liver_thresh* | liver threshold |

**Returns**

New labelled mask

Steps:

- Sets thresholds for current slice
- Runs self.find_organs on current slice with new thresholds

#### 6.4.3.16 writePillow2PNG()

```
def writePillow2PNG (
              img,
              outfile )  [static]
```

### 6.4.4 Member Data Documentation

#### 6.4.4.1 last_clicked_x

```
last_clicked_x
```

The x-coordinate of the region that was last selected/clicked on.

#### 6.4.4.2 last_clicked_y

```
last_clicked_y
```

The y-coordinate of the region that was last selected/clicked on.

#### 6.4.4.3 LIVER_LABEL

```
int LIVER_LABEL = 1  [static]
```

Regions in mask with this label value are considered liver.

#### 6.4.4.4 masks

`masks`

Dictionary: key: mask index, value: mask numpy array dim(n,m,1)

#### 6.4.4.5 pathToSlices

`pathToSlices`

List contaning the path to the png slice files

#### 6.4.4.6 patient_id

`patient_id`

ID of the patient.

#### 6.4.4.7 selected_pixel_value_of_label_mask

`selected_pixel_value_of_label_mask`

The label of the region that was last selected/clicked on.

#### 6.4.4.8 sliceCache

`sliceCache`

Dictionary: key: slice index, value: slice PNG.

#### 6.4.4.9 SPLEEN_LABEL

`int SPLEEN_LABEL = 2 [static]`

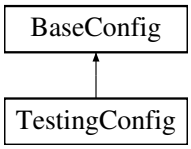
Regions in mask with this label value are considered spleen.

The documentation for this class was generated from the following file:

- /Users/lorenz/Desktop/temp4/radipop/RadiPOP-standalone/web_app/utility/radipop_gui.py

## 6.5 TestingConfig Class Reference

Inheritance diagram for TestingConfig:




### Static Public Attributes

- bool DEBUG = True

### 6.5.1 Member Data Documentation

#### 6.5.1.1 DEBUG

```
bool DEBUG = True  [static]
```

The documentation for this class was generated from the following file:

- /Users/lorenz/Desktop/temp4/radipop/RadiPOP-standalone/web_app/utility/config.py

# Chapter 7

# File Documentation

## 7.1 /Users/lorenz/Desktop/temp4/radipop/RadiPOP-standalone/web_app/segmenter_flask_API.py File Reference

### Namespaces

- namespace segmenter_flask_API

### Functions

- def correctPartition ()

  *Reveives index to slice/mask + coordinates--> returns partion corrected mask as PNG to client.*
- def dcm2png ()

  *Receive Paths to dcm files, converts them to PNG.*
- def drawOnMask ()

  *Reveives index to slice/mask + x,y coordinates --> returns drawn on mask as PNG to client.*
- def extendThresholds ()

  *Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.*
- def getMask ()

  *Reveives index to slice/mask --> returns mask stored on flask server as PNG to client.*
- def highlightOrgan ()

  *Reveives index of slice + x,y coordinates --> returns highlighted mask as PNG to client.*
- def initialize ()

  *Receive Paths to ordered slices, caches slices.*
- def labelOrgan ()

  *Reveives index to slice mask + label id --> returns highlighted mask as PNG to client.*
- def postPickleGetMask ()

  *Receives path to pickle file --> returns mask as PNG to client.*
- def saveMasks ()

  *Reveives path, saves all stored masks as pickle files to path --> returns output path.*
- def updateMask ()

  *Receives index of slice + slider values --> returns updated mask as PNG to client.*

### Variables

- app = Flask(__name__)
- int FLASK_PORT = 4041
- string FLAST_HOST = '0.0.0.0'
- host
- dictionary patients = {}

  *Dictionary which will hold for each patientID a RadiPopGUI object.*
- port

## 7.2 /Users/lorenz/Desktop/temp4/radipop/RadiPOP-standalone/web_↩ app/utility/config.py File Reference

### Classes

- class BaseConfig
- class DevelopmentConfig
- class ProductionConfig
- class TestingConfig

### Namespaces

- namespace config

## 7.3 /Users/lorenz/Desktop/temp4/radipop/RadiPOP-standalone/web_↩ app/utility/radipop_gui.py File Reference

### Classes

- class RadiPopGUI

  *Bridge between the flask server/API and the RadiPOP segmenter:*

### Namespaces

- namespace radipop_gui

## 7.4 /Users/lorenz/Desktop/temp4/radipop/RadiPOP-standalone/web_↩ app/utility/segmentation_utils.py File Reference

### Namespaces

- namespace segmentation_utils

## Functions

- def add_sobel_edges (mask, img)

  *Smooth edges Steps:*
- def draw_region_outlines (mask)

  *Color the mask light green.*
- def guess_bounds (regions_map, reference_map)

  *Guess the bounds/labels of the region based on reference region Guess the bounds/labels of the region based on reference region (generally neighboring slice).*
- def partition_at_threshold (img, thresh, square_size, min_size, title=None, show_plot=True)

  *After some smoothing, calculate new mask for img Steps:*
- def save_partition (mask, path)

# Index