OrationesPython Documentation

Release 1.0.1

Timo Mätäsaho

November 04, 2013

CONTENTS

Contents:

CHAPTER

ONE

DOCUMENTATION FOR THE CODE

1.1 Orationes Search

osearch.**osearch**(*img*, *txtf*, *sw*)

The main program that only calls the processing methods from OratUtils and HFun classes.

Parameters

- **img** (*string*) – The name of the image.
- **txtf** (*string*) – The name of the cleaned XML file.
- **sw** (*string*) – The word or letter that is searched.

Returns nothing or JSON string

The return option have to be chosen between returning the string as a return code or is it just printed out for the calling PHP program. Currently it is being printed.

1.2 OratUtils

class osearch.**OratUtils**

This class contains only static utility methods that are called directly from the main program 'osearch.py'.

static boundingBox(*image*, *debug*)

This functions tries to determine the bounding boxes for each text line.

Parameters

- **image** (*ndarray*) – the processed image
- **debug** (*bool*) – debug switch

Returns ndarray – bboxes

$$bboxes_{nxm} = \begin{bmatrix} \text{patch label numbers} \\ \text{starting x-coordinates} \\ \text{starting y-coordinates} \\ \text{ending x-coordinates} \\ \text{ending y-coordinates} \end{bmatrix}$$

debug switch can be used to plot the results of the bounding box founding method and to see whether it is working correctly.

Process pipeline:

1.Calculate the histogram from the image

2.Binarize image with threshold 0.95

3.Label all the patched in on the binarized image

4.Calculate the sizes of the patches

5.Remove unnecessary patches

(a)Remove the largest patch. The largest patch is always the patch consisting of the borders and marginals.

(b)Remove patches which size is smaller or equal to 50 pixels

(c)Remove all the patches which are higher than 70 pixels. This removes the possible remaining marginal patches which weren't connected to the major marginal and border patch.

6.Perform morpholig operations to clean the image and bind the text lines together

(a)Perform erosion with a cross like structure element

$$SEe_{5,5} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

(b)Perform dilation with a long vertical line. (needs a 70x70 size structure element)

$$SEd_{70,70} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ & \vdots & & \vdots & \\ 1 & 1 & \dots & 1 & 1 \\ & \vdots & & \vdots & \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

7.Label the morphologically operated image

8.Remove patches which size is less or equal to 4000 pixels

9.Label the image again with new labels

10.Calculate the extreme dimensions of each patch. These values are used as the limiting bounding boxes.

11.Combine the boxes which are horizontally too close as they are thought to be separate boxes on the same textline.

12.Return the bounding boxes

static `contStretch`(*im*, *a*, *h*)

Performs contrast stretching for grayscale images. Pixel intensities are set to differ 'a' times the average

intensity from the original intensity values. The new intensity values are sliced to stay between [0, 255].

$$I_{stretched} = I_{old} + a * (I_{old} - I_{average})$$
$$I_{new} = \begin{cases} 0, & I_{stretched} < 0 \\ I_{stretched}, & 0 \leq I_{stretched} \leq 255 \\ 255, & I_{stretched} > 255 \end{cases}$$

Parameters

- **im** (*ndarray*) – The image which contrast is to be stretched
- **a** (*int*) – multiplication coefficient
- **h** (*int*) – The height of image. Used as partial image average switch

Returns ndarray – contrast stretched image

Parameter *h* is a switch which could be used to determine if the average intensity is calculated over the whole image or from a small portion of it. Currently it is defaulted in the code to newer happen. Originally the idea was that if the image is very big, the intensity average would be taken from a small sample. To make the function more generic and also because of the nature of the images in Orationes project, it was decided that the average is always calculated over the whole image.

static `hfilter` (*image*, *diameter*, *height*, *length*, *n*)

This function performs homomorphic filtering on grayscale images.

Parameters

- **image** (*ndarray*) – 2-dimensional ndarray
- **diameter** (*int*) – filter diameter
- **height** (*int*) – Height of the image
- **length** (*int*) – Length of the image
- **n** (*int*) – Filter order

Returns ndarray – homomorphically filtered image

The image must in ndarray format. In osearch PIL images are converted to scipy images which are in ndarray format. Ndarray format allows easy and fast direct access to the pixel values and this function is written entirely only for the ndarrays.

static `padlines` (*imlines*, *llines*, *charlines*)

Parameters

- **imlines** (*ndarray*) – n*1 size ndarray containing the lines (or rather their y-position) got from the image by radontransform
- **llines** (*ndarray*) – n*2 size ndarray containing the length information of the lines
- **charlines** (*list*) – list of lists telling the position(s) of searched character(s)/word(s) on each line

Returns ndarray – wantedlines

Long: Llines contains the information about the lines got from the XML and also it contains the information if some of the lines are remarkably shorter than other lines. That means that, if there are some lines that are not found from the image, it is assumed that those non-found lines are the shortest lines according to the XML and character count. Those lines are marked as 0 in the second column in llines.

Short: Llines[:,1] contains only 1s and 0s. 1 meaning a line with enough letters to be recognized by poormanradon (pmr) and 0 meaning a line which is probably undetected by pmr

Behavior: Number of lines found from the image using pmr is larger than the number of lines calculated from XML:

TODO! Currently this case is not handled!

Number of lines found from the image using pmr is smaller than the number of lines calculated from XML:

Pad the lines according to the information in llines[:,1]

llines: imlines: rlines:

[1 ———->[100 ——>[100 1 ———-> 200 ——> 200 1 ———-> 300 ——> 300 1 ———-> 400 ——> 400 0 .——> 600 ——. NAN 1 /.——> 700 ——.‘600 1 / .——-> 900]——-. ‘700 0 / NAN 1]/ ‘900]

Number of lines found from the image using pmr equals to the number of lines calculated from XML:

Pick unique lines from imlines and return them as lines the interesting lines.

Jos löydettyjä rivejä on vähemmän kuin xml:ssä rivejä, pitää rivejä # tasata ja niiden indeksejä vastaamaan mahdollisimman paljon oikeita # rivejä. Oletuksena on, että rivit, joissa on keskimääräistä vähemmän # kirjiamia, ei tunnistu poormanradonissa, joten ne jää välistä pois ja ne # hylätään kokonaan. Tätä oletusta hyväksi käyttäen kuitenkin korjataan # rivien indeksit osoittamaan aina oikeaan riviin.

static `poormanradon` (*image*, *iname*, *height*, *debug*)

Performs a naive radon-transform on the binarized and contrast stretched image and tries to determine where the text lines are in the image

Parameters

- **image** (*ndarray*) – Image
- **iname** (*string*) – Image name
- **height** (*int*) – Image height
- **debug** (*bool*) – Debug switch

Returns ndarray – Array containing the lines which are found using radon transform

static `stringparser` (*tfile*, *c*)

Performs case sensitive search for text file tfile with string or character c (char on default). Argument c can be any regular expression

Parameters

- **tfile** (*string*) – The name of the cleaned XML file
- **c** (*string/char/regular expression*) – The letter or string that is searched from the tfile

Returns list – charcount

Returns list of lists – charpos

Returns list of lists – charlines

Returns list of lists – wordlens

•*Charcount* is a list containing the lengths of each line.

–`[63, 60, 4, 65, 66, 37, 66, ...]`

•*Charpos* is a list containing lists including the positions of the found characters or the first letters of the found words.

–`[[52], [10, 47, 62], [19, 62], [51], ...]`

- *Charlines* is a list of lists where the length of each sublist tells the number of hits on that line and the element values representing the line number from the XML file.
  - `[[3], [4, 4, 4], [6, 6], [7], ...]`
- *Wordlens* is a list containing lists containing the lengths of the words on each line.
  - `[[3], [3, 3, 3], [3, 3], [3], ...]`

1.3 Helper Functions

class `osearch.`**`HFun`**

CHAPTER

TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

INDEX