Marcin Nawrocki, Tomasz Jankowski

Dokumentacja biblioteki przetwarzania obrazu

Projekt POiSA

Poznań
31.05.2020

Spis treści

api.py

FUNCTIONS

addGaussianNoise(filename, std_dev=0.05, mean=0, number_of_inters=1)

Adding gaussian noise with to given image

Keyword argument:

filename -- filename or path to image

std_dev -- standard deviation parameter

mean -- mean parameter (default = 0)

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful.

addSaltPepperNoise(filename, propability=0.05, saltPepperRatio=0.5, number_of_inters=1)

Adding salt pepper noise to given image

Keyword argument:

filename -- filename or path to image

propability -- how much of image should be noising. Propability that single pixel become salt/pepper noise.

(default = 0.5)

saltPepperRatio -- specified salt to pepper ratio (default 0.5):

1.0 -- only salt

0.5 -- equal propability of salt and pepper

0.0 -- only pepper

Return:

1 if operation was succesful.

filteringImage(filename, np_mask_pom, number_of_inters=1)

Processing filtering with given kernel.

If RGB image is passed, then each channel will be filter separately.

Keyword argument:

filename -- filename or path to image

np_mask -- mask matrix as numpy array

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful.

gammaCorrection(filename, gamma, number_of_inters=1)

Processing gamma correction with specified gamma correction atribute

If RGB image is passed, then each channel will be correct separately.

Keyword argument:

filename -- filename or path to image

gamma - gamma correction parameter

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful.

getBinaryzedImage(filename, threshold, number_of_inters=1)

Return binaryzed image based on threshold given by user

Keyword argument:

filename -- filename or path to image

thershold -- integer value in range (0,255)

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful

0 if threshold is out of range

getClosely(filename, struct_elem='rect', size=3, number_of_inters=4)

Execute openly(erode and dilate on the same image) morphological operation on image

Function binaryzed image by OTSU method, so pass RGB or grayscale images possible.

Keyword argument:

filename -- filename or path to image

struct_elem:

cross - cross structural element

rect - rectangle structural element

circ -- cricle structural element(maybe implemente)

size: size of struct element, should be 2N+1

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful.

getDilate(filename, struct_elem='rect', size=3, number_of_inters=2)

Execute dilate morphological operation on image

Function binaryzed image by OTSU method, so pass RGB or grayscale images possible.

Keyword argument:

filename -- filename or path to image

struct_elem:

cross - cross structural element

rect - rectangle structural element

circ -- cricle structural element(maybe implemente)

size-- size of struct element, should be 2N+1

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful.

getErode(filename, struct_elem='cross', size=3, number_of_inters=2)

Execute erode morphological operation on image

Function binaryzed image by OTSU method, so pass RGB or grayscale images possible.

Keyword argument:

filename -- filename or path to image

struct_elem:

cross - cross structural element

rect - rectangle structural element

circ -- cricle structural element(maybe implemente)

size: size of struct element, should be 2N+1

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful.

getImageParameters(filename, save_source=True, defaultImagePath='./public/python/images/')

Reading image from file and save parameters in the json file

Keyword argument:

filename -- relative path to image

Return:

image as Numpy Array

getOpenly(filename, struct_elem='rect', size=3, number_of_inters=3)

Execute openly(erode and dilate on the same image) morphological operation on image

Function binaryzed image by OTSU method, so pass RGB or grayscale images possible.

Keyword argument:

filename -- filename or path to image

struct_elem:

cross - cross structural element

rect - rectangle structural element

circ -- cricle structural element(maybe implemente)

size: size of struct element, should be 2N+1

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful.

getOtsuBinaryzedImage(filename, number_of_inters=1)

Return binaryzed image, by threshold which is generated by OTSU method.

Algorithm calculated Otsu using maximalization between class variance.

Keyword argument:

filename -- filename or path to image

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful.

medianFiltergingImage(filename, struct_elem='rect', size=3, number_of_inters=1)

Processing median filtering with specified shape and size on image given by filename

If RGB image is passed, then each channel will be filter separately.

Keyword argument:

filename -- filename or path to image

struct_elem:

cross -- cross structural element

rect -- rectangle structural element

circ -- cricle structural element(maybe will be implemented)

size: size of struct element, should be 2N+1

number_of_inters -- number of inter images which should be generated by this operation

Return:

1 if operation was succesful.

removeFiles()

Function used to delete generated images

toGrayscale(filename, gray='human')

Convert image to grayscale

Keyword argument:

filename -- filename or path to image

gray:

human - convert to "human" grayscale (0,215*R+0.7151*G+0.0721*B)

machine - convert to "machine" grayscale (R+G+B)/3

Return:

Important:

!!! This operation reducing Array dimension from 3 to 2 !!!

binary_operations.py

FUNCTIONS

dilate(np_image_bin, struct_elem='rect', size=3)

Execute dilate morphological operation on binaryzed image

Keyword argument:

np_image_bin -- binaryzed image

struct_elem:

cross - cross structural element

rect - rectangle structural element

circ -- cricle structural element(maybe implemente)

size: size of struct element, should be 2N+1

Return:

Binarized image after dilatation operation

erode(np_image_bin, struct_elem='rect', size=3)

Execute erode morphological operation on binaryzed image

Keyword argument:

np_image_bin -- binaryzed image

struct_elem:

cross - cross structural element

rect - rectangle structural element

circ -- cricle structural element(maybe implemente)

size: size of struct element, should be 2N+1

Return:

Binarized image after erode operation

otsuBinaryzation(np_image_2D)

Return binaryzed image, getting by use of Otsu method

Algorithm calculated Otsu using maximalization between class variance

Keyword argument:

np_image_2D -- image as NumPy array

Return:

Binaryzed image as numpy array with 0 and 255 values

thresholdBinaryzation(np_image_2D, threshold)

Return binaryzed image based on threshold given by user

Keyword argument:

np_image_2D -- image as NumPy array

thershold -- integer value in range (0,255)

Return:

Binaryzed image as numpy array with 0 and 255 values

Help on module basic_operations in basic_operations:

basic_operations.py

FUNCTIONS

convert(o)

Convert function, needed to dump numpy datatypes into JSON file

ensure3D(np_image)

Ensures that given image is 3 dimensional. If is 2D(grayscale) change to 3D(this operation do not changes information in image)

e.g np_image with shape(x,y) will be reshaping to (x,y,1)

ensureGrayscale(np_image, info=False)

Ensures that given image is in grayscale

Keyword argument:

np_image -- image as NumPy array

Return:

np_image as grayscale image

generateInterImages(np_source, np_final, number_of_inters, start_image_number=0, defaultImagePath='./public/python/images/')

Function generates specified number of inter images and final images, then saved them in specified locations,

with filenames as numbers. Real number of generated inter images is:

number_of_inters - start_image number

Additionally final image, will be saved.

When number_of_inters == start_image_number, only final image will be saved

Keyword argument:

np_source -- based image

np_final -- final image

number_of_inters -- specified number of inter images

start_image_number -- number, which starts saving images

defaultImagePath -- path to saving image

getHumanGrayscale(np_image)

Convert image to "Human" grayscale (0,215*R+0.7151*G+0.0721*B)

Keyword argument:

np_image -- image as NumPy array

Return:

np_image_gray -- image as grayscale

Important:

!!! This operation reducing Array dimension from 3 to 2 !!!

getImageColorType(np_image)

Return image mode

Keyword argument:

np_image -- image as NumPy array

Return:

String data:

L for grayscale image

RGB for color image

getImageHistogram(np_image_2dim, normalize=False, with_bins=False)

Return histogram for image in 2D Numpy array(grayscale or single channel)

Keyword argument:

np_image_2dim -- image as 2D NumPy array(whole grayscale or one color channel)

normalize -- if set to True histogram values will be normalized(default = False)

with_bins -- return also bins as numpy arra(default= False)

Return:

histogram as NumPy array,

bins as NumPy array if with_bins = True

getMachineGrayscale(np_image)

Convert image to "Machine" grayscale (R+G+B)/3

Keyword argument:

np_image -- image as NumPy array

Return:

np_image_gray -- image as grayscale

Important:

!!! This operation reducing Array dimension from 3 to 2 !!!

getMinMaxPix(np_image)

Return dictionary with max and min pixel value

Keyword argument:

np_image -- image as NumPy array

Return:

Python dictionary with keys: Max value, Min value

getStatisticImageParameters(np_image)

Return statistical image parameters as dictionary(Variance, Standard devation, Median, Average)

Keyword argument:

np_image_2dim -- image as 2D NumPy array(whole grayscale or one color channel)

Return:

Python dictionary with keys: Variance, Standard devation, Median, Average

getWindow(np_image_bin, index, dir_size, struct_elem)

Get window for morphological and filtering operations

Keyword argument:

index -- indexes of actual processing pixel as tuple

dir_size -- size of structural element in one direction (dir_size = (size-1)/2)

x_max -- max value of x index

y_max -- max value of y index

Return:

np_window -- window of morphological operations with specific size and shape

glueImage(splitted)

Glued vertically splitted images into one image.

Keyword argument:

splitted -- list of images as numpy arrays

Return

np_glued -- new, glued image as numpy array

grayTo2D(np_image)

Reshaping np_image grayscale image to 2 dimension

e.g np_image with shape(x,y,1) will be reshaping to (x,y)

isColorImage(np_image)

Check if image is colored (has 3 channels)

Return

True if image is colored, false otherwise

readImage(filename, verbose=False)

Reading image from file and transform it to NumPy array

Keyword argument:

filename -- relative path to image

verbose -- if true showing image(default False)

Return:

image as Numpy Array

saveImage(np_image, filename, verbose=False)

Saving image to file

Keyword argument:

np_image -- image as NumPy array

filename -- relative to where image should be saved with filename conatining extension

verbose -- if true showing image(default False)

Return:

---

splitImage(np_image, number_of_parts)

Splitting image as 2-dimensional numpy array into array of parts.

Keyword argument:

np_image -- image to split, works for 2D and 3D images

number_of_parts -- number of parts to split image

Return

np_split -- list of numpy array f.e:

1,2,3,4 is parts of image as numpy array, function return list [1,2,3,4]

filtering.py

FUNCTIONS

gammaCorrection(np_image_2D, gamma)

Processing gamma correction with specified gamma correction atribute

Keyword argument:

np_image_2D -- two dimensional image(grayscale or single color channel)

gamma - gamma correction parameter

Return:

np_image_gamma -- image as numpy 2D array, after gamma correction

matrixFilter(np_image_2D, np_mask)

Processing filtering with given matrix

Keyword argument:

np_image_2D -- two dimensional image(grayscale or single color channel)

np_mask -- mask matrix as numpy array

Return:

np_image_fil -- image as numpy 2D array, after specified filtering

medianFilter(np_image_2D, struct_elem='rect', size=3)

Processing median filtering with specified shape and size on given 2 dimensional image

Keyword argument:

np_image_2D -- two dimensional image(grayscale or single color channel)

struct_elem:

cross -- cross structural element

rect -- rectangle structural element

circ -- cricle structural element(maybe will be implemented)

size: size of struct element, should be 2N+1

Return:

np_image_fil -- image as numpy 2D array, after median filtering

noising.py

FUNCTIONS

gaussianNoise(np_image_3D, std_dev=0.1, mean=0)

Adding gaussian noise with to given image

Keyword argument:

np_image_3D -- image with 3 dimensions to apply noising

std_dev -- standard deviation parameter

mean -- mean parameter (default = 0)

Return:

Image with gaussian noise specified by parameters. Dimension the same as given.

saltPepperNoising(np_image, propability=0.05, saltPepperRatio=0.5)

Adding salt pepper nois to given image

Keyword argument:

np_image -- image to apply noising

propability -- how much of image should be noising. Propability that single pixel become salt/pepper noise.

(default = 0.5)

saltPepperRatio -- specified salt to pepper ratio (default 0.5):

1.0 -- only salt

0.5 -- equal propability of salt and pepper

0.0 -- only pepper

Return:

Image noised with specified values. Dimension the same as given.