SZCZECIN UNIVERSITY OF TECHNOLOGY

Automatic Defect Detection and Identification Processor
ADDIP

USER MANUAL

(modified May, 6th 2009)

May, 2007

Index

Introduction

Automatic Defect Detection and Identification Processor (ADDIP) is a collection of advanced image processing algorithms dedicated for automatic radiograph analysis. The ADDIP was created as a programming environment for quick and easy testing newly developed algorithms for defect identification and recognition. Generally, ADDIP is an interpreter of a scripting language that currently supports over 50 commands that can operate on various data types (images, matrices etc). The one of the advantages of ADDIP is that the code of the processor is completely independent from the codes of the implemented (in it) functions. Thanks to this a new function can be added to the processor within very short time and existing function can be easily transferred to an another project with minimal changes in their code.

Here are some basic rules of correct writing a script files:

1. Only one instruction per line is allowed
2. Comments can't be in the same line as commands
3. Particular line can't be longer than 256 chars
4. Use double backslash in the file paths
5. A line should not begin with "white chars" (spaces, tabs…)

Supported datatypes:

- IMAGE
- STRING
- MATRIX
- NUMBER

Known Bugs:

- using variable names the same as the function names may crash the program
- sometimes ADDIP do not load external DLLs if script is run by double click

ADD

Syntax:							
Parameters:							
Description:	Adds two numbers.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	*result* = **add**(*number1*,*number2*);						
Script parameters:	*number1*– number to add *number2*– number to add *result* – result						

Example script:	*output = add(4, 5);*
	% result is saved to res.txt file
	matrixwrite(output,'c:\\res.txt');						

Remarks:	none						

ANNclassifier

<table>
<tr><td>**Syntax:**</td><td colspan="7"></td></tr>
<tr><td>**Parameters:**</td><td colspan="7"></td></tr>
<tr><td>**Description:**</td><td colspan="7">Performs Artificial Neural Network (ANN) based flaw's classification. Operates on tables of features (input vectors arranged in column wise order). Output table consists of classification vectors arranged also in column wise order. Position of the biggest value in output vector determine the class number.</td></tr>
<tr><td>**References:**</td><td colspan="7">[1] Gang Wang, T. Warren Liao: "*Automatic identification of different types of welding defects in radiographic images*", NDT&E International 35 (2002), p. 519-528
[2] Shaun W. Lawson, Graham A. Parker: "*Intelligent segmentation of industrial radiographic images using neural networks*", Proc. SPIE Vol. 2347, p. 245-255, Machine Vision Applications, Architectures, and Systems Integration III, 1994</td></tr>
<tr><td>**List of used functions:**</td><td colspan="7"></td></tr>
<tr><td>**File name (C-code):**</td><td colspan="2"></td><td>**DLL – library name:**</td><td colspan="4"></td></tr>
<tr><td>**Developed by:**</td><td>PSz</td><td>**Date of development:**</td><td>2007.08.1</td><td>**Date of last update:**</td><td>2007.08.1</td><td>**Version:**</td><td>1.0</td></tr>
<tr><td>**History of updates:**</td><td colspan="7">none</td></tr>
<tr><td>**Script syntax:**</td><td colspan="7">*output* = **ANNclassifier** (*input*, '*filename*');</td></tr>
<tr><td>**Script parameters:**</td><td colspan="7">*input* - input table of features
output - output table of features
'*filename*' - name of the file where the ANN structure is stored</td></tr>
<tr><td>**Example script:**</td><td colspan="7">chdir('C:\\addip\radiograms');
% view of input vector stored in inpvector.txt file:
% 2
% 4
% [0,0,1,1;
% 0,1,0,1]
% reading the features table from file
in = matrixread('*inpvector.txt*');
% performing classification process (XOR function)
out = **ANNclassifier**(*in*,'*ffnet_xor_NN_classifier_10_5_2.ann*');
% view of output vector stored in outvector.txt file
% 2
% 4
% [0.998627, 0.001739, 0.001430, 0.998425;
% 0.002549, 0.999784, 0.999815, 0.002774]
% ^ ^ ^ ^
% class1 class2 class2 class1
% writing results to file
matrixwrite(*out*,'*outvector.txt*');</td></tr>
<tr><td>**Remarks:**</td><td colspan="7">none</td></tr>
</table>

ANNCLS

<table>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Performs Artificial Neural Network (ANN) based flaw's classification. Operates on normalized tables of features (input vectors arranged in row wise order). Output table consists of classification vectors arranged in column wise order.</td></tr>
<tr><td><b>References:</b></td><td colspan="7">[1] Gang Wang, T. Warren Liao: "<i>Automatic identification of different types of welding defects in radiographic images</i>", NDT&E International 35 (2002), p. 519-528

[2] Shaun W. Lawson, Graham A. Parker: "<i>Intelligent segmentation of industrial radiographic images using neural networks</i>", Proc. SPIE Vol. 2347, p. 245-255, Machine Vision Applications, Architectures, and Systems Integration III, 1994</td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7"></td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3">ANNcls.dll</td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.08.1</td><td><b>Date of last update:</b></td><td>2007.10.2</td><td><b>Version:</b></td><td>1.2</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7">[<i>out</i>, <i>disc</i>] = <b>ANNcls(</b><i>tab</i>, '<i>filename</i>', <i>num</i>);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>tab</i> - input table of features
'<i>filename</i>' - name of the *.ann file where the ANN structure is stored
<i>num</i> - number of defects
<i>out</i> - table of ANN outputs
<i>disc</i> - classes vector</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">chdir('C:\\addip\\radiograms');
% read the image
original = imread('Flaw_mix_02.tif');
cut = cutroi(original,600,2000,500,1100);
% and display it on the screen (without scaling)
handle1 = imshow('original',cut,0);
% median filtering
med = fastmedian(cut,0,71,71);
% substracting filtered image from original image
sub = imsub(cut,med);
handle2 = imshow('sub',sub,1);
% iterative thresholding
th = ith(sub,0.85);
% removing small dots
out_med = fastmedian(th,0,7,7);
% recognizing flaws
indexed_image = index(out_med);
[num,tab] = calcfeat (cut,indexed_image,out_med,30,100000);
savefeat(tab,num,'features.txt');
im_f = genindexedimg(cut,out_med,tab,num,1);
% displaying result
handle3 = imshow('ith',out_med,1);
handle4 = imshow('numbered',im_f,1);
% performing classification process (requires ANNcls.dll)
[out,disc] = ANNcls(tab,'ffnet_round_elong_NN_classifier_4_3_2.ann',num);
% writing results to file
matrixwrite(out,'outvector.txt');
matrixwrite(disc,'outvectorDisc.txt');
%
% exemplary view of 'outvector.txt' (2 classes applied):
% [0.990233, 0.990364, 0.990365, 0.990379, 0.989409;
% 0.008268, 0.008172, 0.008170, 0.008162, 0.008881]
% ^ ^ ^ ^ ^
% defect1 defect2 defect3 defect4 defect5
%
% exemplary view of 'outvectorDisc.txt'(1-round defect, 2-long defect):
% [1.000000, 1.000000, 1.000000, 1.000000, 1.000000]
% ^ ^ ^ ^ ^
% class of class of class of class of class of
% defect1 defect2 defect3 defect4 defect5</td></tr>
</table>

Remarks:	none

ANNthreshold

<table>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Performs flaw’s detection algorithm using Artificial Neural Network (ANN) technique. The filtering window moves from pixel to pixel over an input image. Acquired data (intensities of pixels) are given as an input of previously trained nonlinear neural filter. For each position of the window one point of output image is estimated. Operates on gray scale images. Output image has the same size as input.</td></tr>
<tr><td><b>References:</b></td><td colspan="7">Shaun W. Lawson, Graham A. Parker: “<i>Intelligent segmentation of industrial radiographic images using neural networks</i>”, Proc. SPIE Vol. 2347, p. 245-255, Machine Vision Applications, Architectures, and Systems Integration III, 1994</td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7"></td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.04.12</td><td><b>Date of last update:</b></td><td>2007.04.12</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><i>output</i> = <b>ANNthreshold</b> (<i>input</i>, ‘<i>filename</i>’);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>input</i> - input image
<i>output</i> - output image
‘<i>filename</i>’ - name of the file where the ANN and mask structure are stored</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">chdir('C:\\addip\radiograms');
% reading the image
<i>original</i> = imread('TestSample_2_3.tif');
% displaying it on the screen (without scaling)
<i>handle1</i> = imshow('original',<i>original</i>,0);
% defects detection by means of neural filter
<i>nn_out</i> = <b>ANNthreshold</b>(<i>original</i>,'c:\\addip\\scipts\\crossMask_33x33_ANN_ff_14_7_1.ann');
% removing small dots
<i>out_med</i> = median(<i>nn_out</i>,1,5,5);
% displaying result
<i>handle2</i> = imshow('nn',<i>out_med</i>,1);</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

ATH

Syntax:							
Parameters:							
Description:	Local thresholding; operates on gray scale images; the calculation of threshold is based on discrimination analysis that separate statistically image on two classes, object and background.; output is a binary image						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **ath**(*input, MaskSize, ThCoef*);						
Script parameters:	*Input* - input image; *output* - output image *MaskSize* – size of the moving mask which contains neighborhood of selected image point *ThCoef* – threshold coefficient, that moves threshold value						

Example script:	chdir('C:\\addip\radiograms');
	% using adaptive thresholding
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = median(*a*,0,51,51);
	out_sub = imsub(*a*,*out_med*);
	out_ith = **ath**(*out_sub*,75,1550);
	out_med1 = median(*out_ith*,0,9,9);
	handle1 = imshow('ith',*out_med1*,1);						

Remarks:	none						

BDILATE

Syntax:							
Parameters:							
Description:	Faster version of Dilate function, used with binary images.						
References:	Gonzalez R. C.: „*Digital Image Processing”*, Addison-Wesley Pub (Sd); 3r.e. edition (March 1992), pp. xxx-xxx						
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **bdilate**(*input*, *sizex*, *sizey*);						
Script parameters:	*input* - input image *output* - output image *sizex* - x size of structural element *sizey* - y size of structural element						

Example script:	chdir('C:\\addip\radiograms');
	% using eikiv thresholding
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = median(*a*,0,51,51);
	out_sub = imsub(*a*,*out_med*);
	out_ith = eth(*out_sub*,35,3,10,0.97,128);
	out_med1 = median(*out_ith*,0,9,9);
	out_bdl = **bdilate**(*out_med1*,3,3);
	handle1 = imshow('ith',*out_bdl*,1);						

Remarks:	Structural element dimensions should have odd values. Minimum value is 3.						

BERODE							
Syntax:							
Parameters:							
Description:	Faster version of Erode function, used with binary images.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **berode**(*input*, *sizex*, *sizey*);						
Script parameters:	*input* - input image *output* - output image *sizex* - x size of structural element *sizey* - y size of structural element						

Example script:	chdir('C:\\addip\radiograms');
	% using eikiv thresholding
	a = imread('TestSample_2_3.tif');
	handle = imshow('Original',*a*,0);
	out_med = median(*a*,0,51,51);
	out_sub = imsub(*a*,*out_med*);
	out_ith = eth(out_sub,35,3,10,0.97,128);
	out_med1 = median(*out_ith*,0,9,9);
	out_ber = **berode**(*out_med1*,3,3);
	handle1 = imshow('Binary Erode',*out_ber*,1);						

Remarks:	Structural element dimensions should have odd values. Minimum value is 3.						

CALCFEAT							
Syntax:							
Parameters:							
Description:	Calculating features; output number of classified defects; table with defects features; (new version of the CALCFEATURES)						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**	calcfeat.dll			
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.09.26	**Version:**	1.0
History of updates:	None						
Script syntax:	[*number, tab*] = **calcfeatur**(*inpImage, indImage, binImage, min, max*);						
Script parameters:	*inpImage* - input original image *indImage* - image after function index binImage – binary image *tab* – output table with features *min, max* – min and max size of defects *number* – number of defects						

Example script:	chdir('C:\\');
	% read the image
	original = imread('TestSample_2_3.tif');
	% and display it on the screen (without scaling)
	handle1 = imshow('original',original,0);
	% median filtering
	med = fastmedian(original,51,51);
	% substracting filtered image from original image
	sub = imsub(original,med);
	% iterative thresholding
	th = ith(sub,0.8);
	% removing small dots
	out_med = fastmedian(th,5,5);
	% recognizing flaws
	indexed_image = index(out_med);
	[num,tab] = **calcfeat**(original,indexed_image,out_med,30,100000);
	savefeat(tab,num,'wynik.txt');
	im_f = genindexedimg(original,tab,num);
	% displaying result
	handle2 = imshow('ith',out_med,1);
	handle3 = imshow('numbered',im_f,0);						

Remarks:							

CF

Syntax:							
Parameters:							
Description:	New version of calcfeat. This function supports multithreading and has fixed many errors.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**	cf.dll			
Developed by:	PSz	**Date of development:**	2009.03.20	**Date of last update:**	2009.09.26	**Version:**	1.0
History of updates:	None						
Script syntax:	[*number, tab*] = **calcfeatur**(*org_image*, *indexed_image, binary_image*);						
Script parameters:	*org_image*- input original image *indexed_image*- image after indexing *binary_image* – binary image - after thresholding *tab* – output table with features						

Example script:	threads(4);
	chdir('C:\\');
	org=readtiff('wzorzec.tif');
	bw = readtiff('wzorzec.tif');
	ind = index(*bw*);
	handle3 = imshow('org',*org*,1);
	handle4 = imshow('ind',*ind*,1);
	writetiff(*ind*,'ind.tif',0);
	c = cf(*org*,*ind*,*bw*);
	matrixwrite(*c*,'features_out.m');						

Remarks:							

CHDIR

Syntax:							
Parameters:							
Description:	The chdir function changes the current working directory to the directory specified by *dirname*. The *dirname* parameter must refer to an existing directory.						
References:							
List of used functions:	none						
File name (C-code):		**DLL – library name:**					
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	**chdir**(‘*dirname*’);						
Script parameters:	*dirname* – new directory						

Example script:	% read the image using full path
	original1 = imread('h:\\nn\\TestSample_1_1.tif');
	% set working directory
	chdir('h:\\nn’);
	% then read image using only file name
	original2 = imread('TestSample_1_1.tif');						

Remarks:	This function affects all input/output operations in the whole program (eg, dump, imwrite)						

CROPIMAGE

Syntax:							
Parameters:							
Description:	Crop Image to smaller size. The white area around the weld image is removed.						
References:							
List of used functions:							
File name (C-code):		**DLL – library name:**					
Developed by:	PSz	**Date of development:**	2007.12.3	**Date of last update:**	2008.04.01	**Version:**	4.0
History of updates:	none						
Script syntax:	*output* = cropImage (*input,GrayValueThreshold*);						
Script parameters:	*input* - input image *GrayValueThreshold* – a threshold of gray value, over whose all pixels are croped *output* - cropped output image						

Example script:	chdir('C:\\');
	a = imread('FilmFree_A_Szczecin_NA_NA_NA_TC_Spoiny_27-09-2007_14-40-04_171A.tif');
	handle = imshow('original',a,1);
	b = cropImage(a, 60000);
	handle1 = imshow('croped',b,1);						

Remarks:	none						

<table>
<tr><th colspan="8">CUTROI</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Extracts sub-image laying between *xstart* - *xstop* and *ystart* - *ystop* coordinates from the original image. If extracted region is saved, the *Startx* and *Starty* coordinates are stored in a tiff file in tags TIFFTAG_XPOSITION and TIFFTAG_YPOSITION</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.03.20</td><td><b>Date of last update:</b></td><td>2007.03.20</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7">*output* = **cutroi**(*input, startx, stopx, starty, stopy*);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7">*input – input image*
output – cut output image
startx – beginning x coordinate
stoptx – ending x coordinate
starty – beginning y coordinate
stopy – ending y coordinate</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">*original* = imread('c:\\TestSample_2.3.tif');
handle1 = imshow('original',*original*,0);
cut = **cutroi**(*original*,100,200,100,200);
handle1 = imshow('cut',*cut*,0);</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

DETECTROI

Syntax:							
Parameters:							
Description:	Detects a ROI using gauss function fitted to specified weld profile. By data received from a gauss a middle and edge area of weld are detected. Also heating area is set.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.10.1	**Date of last update:**	2007.10.1	**Version:**	1.0
History of updates:	none						
Script syntax:	*[ReginVectors , MedRegionVector ,output] =* detectROI (*input, dy, Ha, ImRes, Hup, Hdown*, *ErrorLim*, '*filename*');						
Script parameters:	*input* - input image *output* - output image with weld area lines *dy* - leap between to analyzed weld profiles, it specified y size (y size = y image resolution / dy)of the RegionVectors ; *Ha* - heat area (mm) *ImRes* - image resolution (µm) *Hup* - parameter that specifies a position of central weld area (%) *Hdown* - parameter that specifies a position of weld edges *ErrorLim* - limit of error (nor use int this version, set as 0) '*filename*' - name of the file and directory that contains approximation parameters *ReginVectors* – matrix containing a information of weld area lines position for each analyzed line profile [7,y size] *MedRegionVector* - matrix containing a median value of *ReginVectors* for each row						

Example script:	chdir('c:\\Test Welds - Technic Control\\');
	im = imread('FIB94B.TIF');
	
	imcut = cutroi(im,101,2999,101,1899);
	
	handle1 = imshow('imcut',*imcut,1);*
	
	[tab1,tab2,Roi] =
	detectROI*(imcut,*90,10,50,10,20,0,'C:\\ProgramFiles\\PSz\\ADDIP\\ADDIP\\DLL\\ApproximationParam.txt'*);*
	
	handle2 = imshow('Roi',*Roi,0);*
	
	imwrite('c:\\Test Welds - Technic Control\\FIB94B_ROI.tif',Roi);						

Remarks:	none						

DILATE

Syntax:							
Parameters:							
Description:	Dilates input image using rectangular structural element with size specified by user. Operates on grayscale images.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **dilate**(*input*, *sizex*, *sizey*);						
Script parameters:	*input* - input image *output* - output image *sizex* - x size of structural element *sizey* - y size of structural element						

Example script:	chdir('C:\\addip\radiograms');
	a = imread('TestSample_2_3.tif');
	handle = imshow('Original',*a*,0);
	out_dil = **dilate**(*a*,3,3);
	handle1 = imshow('Dilate',*out_dil*,1);						

Remarks:	Structural element dimensions should hale odd values. Minimum value is 3.						

DM

Syntax:							
Parameters:							
Description:	Allows user to input matrixes directly to ADDIP workspace.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.08.03	**Date of last update:**	2007.08.03	**Version:**	0.9
History of updates:	none						
Script syntax:	*matrix* = **dm**(*'matrix_def'*);						
Script parameters:	*matrix – output matrix* *matrix_def – string that defines the matrix*						
Example script:	*Matrix* = **dm**('[1,1,1;2,1,3]'); *% now matrix is 1 1 1* *% 2 1 2*						
Remarks:	Use [] to concentrate numbers into matrix. Commas define columns whereas ; defines next row.						

ERODE

Syntax:							
Parameters:							
Description:	Erodes input image using rectangular structural element with size specified by user. Operates on grayscale images.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **erode**(*input*, *sizex*, *sizey*);						
Script parameters:	*input* - input image *output* - output image *sizex* - x size of structural element *sizey* - y size of structural element						

Example script:	chdir('C:\\addip\radiograms');
	a = imread('TestSample_2_3.tif');
	handle = imshow('Original',*a*,0);
	out_erd = **erode**(*a*,3,3);
	handle1 = imshow('Erode',*out_erd*,1);						

Remarks:	Structural element dimensions should hale odd values. Minimum value is 3.						

ETH

<table>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Local thresholding; thresholding; operates on gray scale images; the threshold is calculated for each area assigned to a small mask, using Otsu Threshold algorithm on data taken from area assigned to large mask; output is a binary image</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.03.20</td><td><b>Date of last update:</b></td><td>2007.03.20</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><i>output</i> = <b>eth</b>(<i>input, LargeMaskSize, SmallMaskSize, minDiff, ThCoef, GrayScalLevels</i>);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>Input</i> - input image;
<i>output</i> - output image
<i>SmallMaskSize</i> – size of small moving mask, that is being thresholded
<i>LargeMaskSize</i> – size of the large moving mask, that contain data to calculate threshold
<i>minDiff</i> – minimum difference between mean value of background and mean value of object information
<i>ThCoef</i> – threshold coefficient, that moves threshold value
<i>GrayScaleLevels</i> – number of gray levels, that image is normalized to</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">chdir('C:\\addip\radiograms');
% using eikiv thresholding
<i>a</i> = imread('TestSample_2_3.tif');
<i>handle</i> = imshow('original',<i>a</i>,0);
<i>out_med</i> = median(<i>a</i>,0,51,51);
<i>out_sub</i> = imsub(<i>a</i>,<i>out_med</i>);
<i>out_ith</i> = <b>eth</b>(<i>out_sub</i>,35,3,10,0.97,128);
<i>out_med1</i> = median(<i>out_ith</i>,0,9,9);
<i>handle1</i> = imshow('ith',<i>out_med1</i>,1);</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

<table>
<tr><th colspan="8">EXIT</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Closes main program and passes error code to the system</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.03.20</td><td><b>Date of last update:</b></td><td>2007.03.20</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><b>exit</b>(<i>code</i>);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>code</i> - system error code (0 – no error)<i>;</i></td></tr>
<tr><td><b>Example script:</b></td><td colspan="7"><b>exit</b>(0)</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

FASTMEDIAN							
Syntax:							
Parameters:							
Description:	2-D Median filter; quick-sort algorithm; operates on gray scale images; output image has the same size as input; In next position of mask one column is sorted, previous data are stored						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.3
History of updates:	none						
Script syntax:	*output* = **fastmedian_**(*input*, *type*, *XmaskSize*, *YmaskSize*);						
Script parameters:	*input* - input image *output* - output image *type* – not used yet *XmaskSize* - x size of a mask *YmaskSize* - y size of a mask						

Example script:	chdir('C:\\addip\radiograms');
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = **fastmedian**(*a*,0,51,51);
	handle1 = imshow('ith',*out_med*,0);						

Remarks:	none						

FASTMEDIANMP

Syntax:	
Parameters:	
Description:	2-D Median filter; quick-sort algorithm; operates on gray scale images; output image has the same size as input; In next position of mask one column is sorted, previous data are stored. This version supports multithreading (up to 2 cores). Especially suitable for images having number of columns >> number of rows
References:	
List of used functions:	none

File name (C-code):		**DLL – library name:**	

Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.3

History of updates:	none
Script syntax:	*output* = **fastmedianmp**(*input*, *type*, *XmaskSize*, *YmaskSize*);
Script parameters:	*input* - input image *output* - output image *type* – not used yet *XmaskSize* - x size of a mask *YmaskSize* - y size of a mask

Example script:	chdir('C:\\addip\radiograms');
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = **fastmedianmp**(*a*,0,51,51);
	handle1 = imshow('ith',*out_med*,0);

Remarks:	none

FFTFILT

Syntax:							
Parameters:							
Description:	Performs a low-pas, band-pass or high-pass filtering on the original image using forward and reverse transform. The cutoff frequency is normalized to 0-1 range, where 1 stands for the higher Fourier coefficient number. The transform resolution always equals the image resolution.						
References:	FFTW library documentation						
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.5
History of updates:	none						
Script syntax:	*[filtered, flipped]* = **fftfilt**(*image*, *cutoff1*, *cutoff2*);						
Script parameters:	*image* – original image *filtered* – filtered image *flipped* – extended image (for test only) *cutoff1* – lower cutoff frequency related to image size (cut-off frequency of hi-pass filter) *cutoff2* – upper cutoff frequency related to image size (cut-off frequency of lo-pass filter)						

Example script:	chdir('C:\\addip\radiograms');
	original = imread*('c:\\TestSample_2_3.tif')*
	ee1 = imshow*('original',original,1);*
	[outlo,zer] = fftfilt*(original,0.1,5);*
	[outhi,zer] = fftfilt*(original,3,500);*
	h = imshow*('lo',outlo,1);*
	h1 = imshow*('extended',zer,1);*
	h2 = imshow*('hir',outhi,1);*
	imwrite*('c:\\out.tif',outlo,1);*						

Remarks:	none						

FILLHOLES

Syntax:							
Parameters:							
Description:	Fills every closed holes in objects. Operates on binary images.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **fillholes**(*input*);						
Script parameters:	*input* - input image *output* - output image						
Example script:							
Remarks:							

GENINDEXEDIMG

Syntax:							
Parameters:							
Description:	Generates image with shown defects; (new version of GENINDEXIMAGE)						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**	genindexedimg.dll			
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.09.26	**Version:**	1.0
History of updates:	None						
Script syntax:	**genindexedimg** *(inpImage, binImage, tab, number, color)*						
Script parameters:	*inpImage* - input original image *binImage* – binary image *tab* – table with features *number* – number of defects *color* – color of the description text and outline						

Example script:	chdir('C:\\');
	% read the image
	original = imread('TestSample_2_3.tif');
	% and display it on the screen (without scaling)
	handle1 = imshow('original',*original*,0);
	% median filtering
	med = fastmedian(*original*,51,51);
	% substracting filtered image from original image
	sub = imsub(*original*,*med*);
	% iterative thresholding
	th = ith(*sub*,0.8);
	% removing small dots
	out_med = fastmedian(*th*,5,5);
	% recognizing flaws
	indexed_image = index(*out_med*);
	[*num*,*tab*] = calcfeat(*original*,*indexed_image*,*out_med*,30,100000);
	savefeat(*tab*,*num*,'wynik.txt');
	im_f = **genindexedimg**(*original*,*tab*,*num*);
	% displaying result
	handle2 = imshow('ith',*out_med*,1);
	handle3 = imshow('numbered',*im_f*,0);						

Remarks:	none						

GETIMSIZE

Syntax:							
Parameters:							
Description:	Returns size of the image						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**		Getimsize.dll		
Developed by:	PSz	**Date of development:**	2007.10.28	**Date of last update:**	2007.10.28	**Version:**	1.0
History of updates:	none						
Script syntax:	[*rows*,*cols*] = **getimsize**(*image*);						
Script parameters:	*image*– input image *rows*– number of rows in *image* *cols*– number of columns in *image*						

Example script:	chdir(*'C:\\addip\radiograms');*
	a = readtiff(*'TestSample_2_3.tif'*);
	[*rows*,*cols*] = **getimsize**(*image*);
	matrixwrite(*row,'row.txt'*);
	matrixwrite(*col,'col.txt'*);						

Remarks:	none						

GETMATRIXELEMENT

Syntax:							
Parameters:							
Description:	Extracts one or more elements from matrix. Elements in matrix are indexed from 0. Function accepts particular number of row and column as well as matrices that contain indexes of elements to cut.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**	Getmatrixelement.dll			
Developed by:	PSz	**Date of development:**	2007.10.28	**Date of last update:**	2007.10.28	**Version:**	1.0
History of updates:	none						
Script syntax:	*out* = **getmatrixelement**(*matrix*, *rows*, *cols*);						
Script parameters:	*matrix* – input matrix *out* – extracted elements from *matrix* *rows* – vector of size1xR with indexes of *matrix* elements to cut. R must be smaller than number of rows in *matrix* *cols* – vector of size1xC with indexes of *matrix* elements to cut. C must be smaller than number of columns in *matrix*						

Example script:	*a* = dm('*[1,2,3,4,5; 6,7,8,9,10]*');
	% extracting one element from first row and second column
	out1 = **getmatrixelement**(a,0,1);
	% extracting three elements from second row
	c = dm('[1,2,3]');
	out2 = getmatrixelement(a,1,c);
	matrixwrite(out1,'out1.txt');
	matrixwrite(out2,'out2.txt');						

Remarks:	none						

GTH

Syntax:							
Parameters:							
Description:	Global thresholding; operates on gray scale images; the threshold is calculated according to the maximum gradient of brightness in x and y direction; output is a binary image						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **gth**(*input, MaskSize, ThCoef*);						
Script parameters:	*input* - input image; *output* - output image with a 0,1 binary information where 1 is object and 0 is a background *MaskSize* - size of the moving mask which contains neighborhood of selected image point (odd) *ThCoef* - threshold coefficient, that moves threshold value						

Example script:	chdir('C:\\addip\radiograms');
	% using gradient thresholding
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = median(*a*,0,51,51);
	out_sub = imsub(*a*,*out_med*);
	out_ith = **gth**(*out_sub*,31,0.75);
	out_med1 = median(*out_ith*,0,9,9);
	handle1 = imshow('ith',*out_med1*,1);						

Remarks:	none						

HIST

Syntax:							
Parameters:							
Description:	Displays histogram of the image in the separate window. All values are multiplied by *mag* before displaying. The output window has a fixed size and can't be resized.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	**hist**(*image*, *mag*);						
Script parameters:	*image* - original image *mag* - magnification						

Example script:	chdir('*C:\\addip\radiograms*');
	% read the image
	original = imread('*TestSample_1_1.tif*');
	% histogram in original scale
	hist(*original,1*);
	% half-sized histogram
	hist(*image,0.5*);						

Remarks:	none						

IMADD

Syntax:							
Parameters:							
Description:	Adds two images. The input images are normalized to 0-65535 range before adding. Result is normalized as well. The *image1* and *image2* must be the same size.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **imadd**(*image1, image1*);						
Script parameters:	*image1* – original image *image2* – original image *output* – sum of image1 and image2						

Example script:	chdir('*C:\\addip\radiograms*');
	% read the image
	original1 = imread('*TestSample_1_1.tif*');
	% read the image
	original2 = imread('*TestSample_1_2.tif*');
	% adding images
	output = **imadd**(*original1, original2*);
	% displaying the result
	handle1 = imshow(‘*adding result', output,1*);						

Remarks:	none						

IMCLOSE

Syntax:							
Parameters:							
Description:	Closes the window identified by *handle*.						
References:	OpenCV Library documentation						
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	**imclose**(*handle*);						
Script parameters:	*handle* – a handle returned by imread function						

Example script:	chdir(*'C:\\addip\radiograms'*);
	% read the image
	original1 = imread(*'TestSample_1_1.tif'*);
	% closing
	imclose*(original1);*
	% to see the window opened by imread one must run the program step-by-step						

Remarks:	Useful only during step-by-step running						

IMMULTI

Syntax:							
Parameters:							
Description:	Multiplies image1 by image1. The input images are normalized to 0-65535 range before multiplying. Result is normalized as well. The image1 and image2 must have the same size.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **immulti**(*image1, image1*);						
Script parameters:	*image1* – original image *image2* – original image *output* – sum of image1 and image2						

Example script:	chdir('*C:\\addip\radiograms*');
	% read the image
	original1 = imread('*TestSample_1_1.tif*');
	% read the image
	original2 = imread('*TestSample_1_2.tif*');
	% multiplying images
	output = **immulti** (*original1, original2*);
	% displaying the result
	handle1 = imshow(‘*multiplication result', output,1*);						

Remarks:	none						

IMNORMALIZE							
Syntax:							
Parameters:							
Description:	Image normalization procedure. It is base on rigidly determine the background gray value and stretching the histogram from that point.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**	imNormalize.dll			
Developed by:	PSz	**Date of development:**	2009.03.13	**Date of last update:**	2009.03.13	**Version:**	1.0
History of updates:	None						

Script syntax:	*reg=dm(‘[x1,y1,x2,y2]’);*
	outimage = **imNormalize**(*inpImage, GrayLvl,reg , deltaIBase, Nmin, randLayout,IBaseShift*);						

Script parameters:	*inpImage* - input original image, *GrayLvl* – maximum gray level value of input image, *reg* - region of interest, that calculates IBase (backgroung value), if reg = [0,0,0,0] the background IBase is calculated according to histogram and deltaIBase, *deltaIBase* – range of determine the IBase value on a histogram , *Nmin* - relative minimum on histogram taken to a consideration , *randLayout* – random percentage edge value of a histogram. All pixels with < Nmin are set on the edge. If randLayout = 0 these pixels are set on a value = Mmin or Nmax, *IBaseShift* – target IBase shifted from IBase, during the stretch operations the IBaseShift reminds in the same place						

Example script:	chdir(C:\Program Files\PSz\ADDIP_FilmFree\ADDIP\Radiograms');
	inIm = readtiff('T_14-32-53_836A.tif');
	reg = dm('[0,0,0,0]');
	outIm=imNormalize(inIm,65535,reg,9,2,0,20000);
	handle0 = imshow('inIm',inIm,0);
	handle1 = imshow('outIm',outIm,0);						

Remarks:							

<table>
<tr><th colspan="8">IMPORTBINARYMATRIX</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Imports matrix from Matlab to ADDIP. Save matrix in Matlab using savebinarymatrix.m script</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.03.20</td><td><b>Date of last update:</b></td><td>2007.03.20</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><i>output</i> = <b>importbinarymatrix</b>(‘<i>matrix.txt</i>’);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>matrix.txt</i> – filename</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7"><i>%</i></td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

<table>
<tr><th colspan="8">IMREAD</th></tr>
<tr><td>Syntax:</td><td colspan="7"></td></tr>
<tr><td>Parameters:</td><td colspan="7"></td></tr>
<tr><td>Description:</td><td colspan="7">Imread reads grayscale or BW image in 16-bit tiff format</td></tr>
<tr><td>References:</td><td colspan="7">Libtiff documentation and internet resources.</td></tr>
<tr><td>List of used functions:</td><td colspan="7">none</td></tr>
<tr><td>File name (C-code):</td><td colspan="2"></td><td colspan="2">DLL – library name:</td><td colspan="3"></td></tr>
<tr><td>Developed by:</td><td>PSz</td><td>Date of development:</td><td>2007.03.20</td><td>Date of last update:</td><td>2007.03.20</td><td>Version:</td><td>1.0</td></tr>
<tr><td>History of updates:</td><td colspan="7">none</td></tr>
<tr><td>Script syntax:</td><td colspan="7">Image = imread(filename);</td></tr>
<tr><td>Script parameters:</td><td colspan="7">filename – path and name of the tiff file;
image – opened image</td></tr>
<tr><td>Example script:</td><td colspan="7">chdir('C:\\addip\radiograms');
% read the image
original = imread('TestSample_2_3.tif');
% and display it on the screen (without scaling)
handle1 = imshow('original',original,0);
% median filtering
med = median(original,0,51,51);
% save result to file
imwrite(‘c:\\test,tif’, med,1);</td></tr>
<tr><td>Remarks:</td><td colspan="7">none</td></tr>
</table>

<table>
<tr><th colspan="8">IMSHOW</th></tr>
<tr><td>Syntax:</td><td colspan="7"></td></tr>
<tr><td>Parameters:</td><td colspan="7"></td></tr>
<tr><td>Description:</td><td colspan="7">Displays image on the screen. The images bigger than 1280x1024 are automatically scaled to this size.</td></tr>
<tr><td>References:</td><td colspan="7">OpenCV Library documentation</td></tr>
<tr><td>List of used functions:</td><td colspan="7">none</td></tr>
<tr><td>File name (C-code):</td><td colspan="2"></td><td colspan="2">DLL – library name:</td><td colspan="3"></td></tr>
<tr><td>Developed by:</td><td>PSz</td><td>Date of development:</td><td>2007.03.20</td><td>Date of last update:</td><td>2007.03.20</td><td>Version:</td><td>1.0</td></tr>
<tr><td>History of updates:</td><td colspan="7">none</td></tr>
<tr><td>Script syntax:</td><td colspan="7"><i>handle</i> = <b>imshow</b>(<i>‘name’, image, norm</i>);</td></tr>
<tr><td>Script parameters:</td><td colspan="7"><i>name</i> – window name displayed in the title bar
<i>image</i> – the image to be displayed
<i>handle</i> – handle to identify the window
<i>norm</i> – data range normalization
0 – without normalization – display image as it is
1 – with normalization – image values are scaled to range 0 - 65535</td></tr>
<tr><td>Example script:</td><td colspan="7">chdir('<i>C:\\addip\radiograms</i>');
<i>% read the image</i>
<i>original1</i> = imread('<i>TestSample_1_1.tif</i>');
<i>% read the binary image</i>
<i>original2</i> = imread('<i>TestSample_1_bin.tif</i>');
<i>% display grayscale image</i>
<i>H1</i> = <b>imshow</b>(<i>‘original1’, original1,0</i>);
<i>% displaying the binary image</i>
<i>H2</i> = <b>imshow</b>(<i>‘original2’, original2,1</i>);</td></tr>
<tr><td>Remarks:</td><td colspan="7">The maximal size opened window is 1280x1024. Bigger images are scaled to this resolution. Screen resolution should be 1280x1024 or bigger. Two windows can’t have the same names. If they have, second window erases first window.</td></tr>
</table>

<table>
<tr><th colspan="8">IMSUB</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Subtracts <i>image2</i> from <i>image1</i>. The input images are normalized to 0-65535 range before subtracting. Result is normalized as well. The image1 and image2 must have the same size</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.03.20</td><td><b>Date of last update:</b></td><td>2007.03.20</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><i>output</i> = <b>imsub</b>(<i>image1, image1</i>);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>image1</i> – original image
<i>image2</i> – original image
<i>output</i> – sum of image1 and image2</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">chdir(<i>'C:\\addip\radiograms'</i>);
<i>% read the image</i>
<i>original1</i> = imread(<i>'TestSample_1_1.tif'</i>);
<i>% read the image</i>
<i>original2</i> = imread(<i>'TestSample_1_2.tif'</i>);
<i>% adding images</i>
<i>output</i> = <b>imsub</b>(<i>original1, original2</i>);
<i>% displaying the result</i>
<i>handle1</i> = imshow(<i>‘substracting result', output,1</i>);</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

<table>
<tr><th colspan="8">IMWRITE</th></tr>
<tr><td>Syntax:</td><td colspan="7"></td></tr>
<tr><td>Parameters:</td><td colspan="7"></td></tr>
<tr><td>Description:</td><td colspan="7">Imwrite saves grayscale or BW image on the disk in 16-bit tiff format</td></tr>
<tr><td>References:</td><td colspan="7">Libtiff documentation and internet resources.</td></tr>
<tr><td>List of used functions:</td><td colspan="7">none</td></tr>
<tr><td>File name (C-code):</td><td colspan="2"></td><td colspan="2">DLL – library name:</td><td colspan="3"></td></tr>
<tr><td>Developed by:</td><td>PSz</td><td>Date of development:</td><td>2007.03.20</td><td>Date of last update:</td><td>2007.03.20</td><td>Version:</td><td>1.0</td></tr>
<tr><td>History of updates:</td><td colspan="7">none</td></tr>
<tr><td>Script syntax:</td><td colspan="7">imwrite(filename, image, normalization);</td></tr>
<tr><td>Script parameters:</td><td colspan="7">filename – path and name of the tiff file;
output – image to save
normalization – 0 – without normalization, 1 –linear normalization to range 0-65535</td></tr>
<tr><td>Example script:</td><td colspan="7">chdir('C:\\addip\radiograms');
% read the image
original = imread('TestSample_2_3.tif');
% and display it on the screen (without scaling)
handle1 = imshow('original',original,0);
% median filtering
med = median(original,0,51,51);
% save result to file
imwrite(‘c:\\test,tif’, med,1);</td></tr>
<tr><td>Remarks:</td><td colspan="7">none</td></tr>
</table>

INDEX

Syntax:							
Parameters:							
Description:	segmentation; operates on binary; output image has the same size as input.						
References:	R. Tadusiewicz and P. Korohoda “Komputerowa analiza i przetwarzanie obrazów”						
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	None						
Script syntax:	*output* = **index**(*input*);						
Script parameters:	*input* - input binary image *output* - output image						

Example script:	chdir('C:\\addip\radiograms');
	% read the image
	original = imread('TestSample_2_3.tif');
	% and display it on the screen (without scaling)
	handle1 = imshow('original',original,0);
	% median filtering
	med = fastmedian(original,51,51);
	% substracting filtered image from original image
	sub = imsub(original,med);
	% iterative thresholding
	th = ith(sub,0.8);
	% removing small dots
	out_med = fastmedian(th,5,5);
	% recognizing flaws
	indexed_image = **index**(out_med);
	[tab,num] = calcfeatures(original,indexed_image,out_med,1,100000);
	savefeatures(tab,num,'wynik.txt');
	im_f = genindexedimage(original,tab,num);
	% displaying result
	handle2 = imshow('ith',out_med,1);
	handle3 = imshow('numbered',im_f,0);						

Remarks:	none						

INVERSE

Syntax:							
Parameters:							
Description:	Inverts color map of input image.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2008.03.03	**Date of last update:**	2008.03.03	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **inverse**(*image1*);						
Script parameters:	*image1* – original image *output* – inverted image						

Example script:	chdir('C:\\addip\radiograms');
	original = readtiff('TestIm2.1.tif');
	ee1 = imshow('original',original,1);
	inv inverse(original);
	h2 = imshow('inv',inv,1);						

Remarks:	none						

ISeeCommand

Syntax:							
Parameters:							
Description:	Sending commands to ISee! via telnet interface. Full list of commands available at: http://www.kb.bam.de/~alex/interfacing-with-ic/						
References:							
List of used functions:	none						
File name (C-code):		**DLL – library name:**					
Developed by:	PSz	**Date of development:**	2009.03.12	**Date of last update:**	2009.03.12	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **ISeeCommand**(*command*);						
Script parameters:	*command* - valid ISee!'s telnet command; *output* - ISee!'s answer						

Example script:	% setting port for communication
	iseeport(23);
	% Creating dataset file
	a = ISeeCommand('savedatasetas c:/set.txt');
	% Extracting image file's name from dataset file
	nazwa = ISeeGetFile('c:/set.txt');
	A = readtiff(nazwa);
	% Extracting first ROI from image basing on dataset file
	A = ISeeGetROI(A, 'c:/set.txt', 1);
	handle = imshow('okno', A, 1);
	writetiff(A, 'C:\\a.tif', 0);
	% Opening modified image in ISee!
	b = ISeeCommand('loadimage c:/a.tif');						

Remarks:	None						

<table>
<tr><th colspan="8">ISeeGetFile</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Extraction of image file’s name from ISee!’s dataset file, along with it’s conversion to ADDIP format.</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2009.03.12</td><td><b>Date of last update:</b></td><td>2009.03.12</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><i>output</i> = <b>ISeeGetFile</b>(fileName);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>fileName</i> - name of ISee!’s dataset file;
<i>output</i> - name of image file retrieved from dataset file and converted to ADDIP format</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">% setting port for communication
iseeport(23);
% Creating dataset file
a = ISeeCommand('savedatasetas c:/set.txt');
% Extracting image file’s name from dataset file
nazwa = ISeeGetFile('c:/set.txt');
A = readtiff(nazwa);
% Extracting first ROI from image basing on dataset file
A = ISeeGetROI(A, 'c:/set.txt', 1);
handle = imshow('okno', A, 1);
writetiff(A, 'C:\\a.tif', 0);
% Opening modified image in ISee!
b = ISeeCommand('loadimage c:/a.tif');</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

<table>
<tr><th colspan="8">ISeeGetROI</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Extraction of desired ISee!’s ROI, given: image, ISee!’s dataset file and ROI’s number.</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2009.03.12</td><td><b>Date of last update:</b></td><td>2009.03.12</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><i>output</i> = <b>ISeeGetROI</b>(<i>image, fileName, roi</i>);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>image</i> - image, from which desired ROI is extracted;
<i>fileName</i> - name of ISee!’s dataset file (containing information about ROI’s)
<i>roi</i> – ROI’s number in ISee!
<i>output</i> – desired part of input image. If ROI with specified number does not exist in dataset file, whole input image is returned instead.</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">% setting port for communication
iseeport(23);
% Creating dataset file
a = ISeeCommand('savedatasetas c:/set.txt');
% Extracting image file’s name from dataset file
nazwa = ISeeGetFile('c:/set.txt');
A = readtiff(nazwa);
% Extracting first ROI from image basing on dataset file
A = ISeeGetROI(A, 'c:/set.txt', 1);
handle = imshow('okno', A, 1);
writetiff(A, 'C:\\a.tif', 0);
% Opening modified image in ISee!
b = ISeeCommand('loadimage c:/a.tif');</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

ISEEPORT							
Syntax:							
Parameters:							
Description:	This function initializes global variable *iseeport*. Global variables are available for all other functions.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2009.01.15	**Date of last update:**		**Version:**	1.0
History of updates:	none						
Script syntax:	*iseeport(port);*						
Script parameters:	*port* – port number						
Example script:							
Remarks:	none						

<table>
<tr><th colspan="8">ITH</th></tr>
<tr><td>Syntax:</td><td colspan="7"></td></tr>
<tr><td>Parameters:</td><td colspan="7"></td></tr>
<tr><td>Description:</td><td colspan="7">Global thresholding; operates on gray scale images; the threshold is calculated using bimodal distribution of image histogram.; output is a binary image</td></tr>
<tr><td>References:</td><td colspan="7"></td></tr>
<tr><td>List of used functions:</td><td colspan="7">none</td></tr>
<tr><td>File name (C-code):</td><td colspan="2"></td><td colspan="2">DLL – library name:</td><td colspan="3"></td></tr>
<tr><td>Developed by:</td><td>PSz</td><td>Date of development:</td><td>2007.03.20</td><td>Date of last update:</td><td>2007.03.20</td><td>Version:</td><td>1.0</td></tr>
<tr><td>History of updates:</td><td colspan="7">none</td></tr>
<tr><td>Script syntax:</td><td colspan="7">output = ith(input, ThCoef);</td></tr>
<tr><td>Script parameters:</td><td colspan="7">input - input image;
output - output image with a 0,1 binary information where 1 is object and 0 is a background
ThCoef - threshold coefficient, that moves threshold value</td></tr>
<tr><td>Example script:</td><td colspan="7">chdir('C:\\addip\radiograms');
% using iterative thresholding
a = imread('TestSample_2_3.tif');
handle = imshow('original',a,0);
out_med = median(a,0,51,51);
out_sub = imsub(a,out_med);
out_ith = ith(out_sub,0.8);
out_med1 = median(out_ith,0,9,9);
handle1 = imshow('ith',out_med1,1);</td></tr>
<tr><td>Remarks:</td><td colspan="7">none</td></tr>
</table>

IQIDET

Syntax:							
Parameters:							
Description:	Improves detection of IQIs						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**		iqidet.dll		
Developed by:	PSz	**Date of development:**	2007.10.01	**Date of last update:**	2007.10.01	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **iqidet**(*input*,*angle*,*width*,*num*);						
Script parameters:	*input* - input image *output* - output image *angle* - maximum predicted angle of IQI *width* - width of moving mask *num* - width of detecting strip						

Example script:	chdir('C:\\addip\radiograms');
	% read image
	a = readtiff('IQI_02.tif');
	handle = imshow('original',*a*,1);
	% IQI detection
	iqi = iqidet(a,10,7,3);
	handle1 = imshow('iqi',*iqi*,0);						

Remarks:	Width and num parameter must be an odd values. Effective angle can be set from range <-15;15> degrees. Num parameter must be smaller than width. Function parameters deployed in iqi_out.txt file in working directory.						

IQIMAX

Syntax:							
Parameters:							
Description:	IQImax function find a maximum value of IQIs. It calcultates a median value of every visible wire and its position. This function alsow calculates SNR for each wire.						
References:							
List of used functions:							
File name (C-code):		**DLL – library name:**					
Developed by:	PSz	**Date of development:**	2007.10.7	**Date of last update:**	2007.10.7	**Version:**	2.0
History of updates:	none						
Script syntax:	*output* = IQImax (*inpu_bef,input_aft,Resolution,WireNr,VisibleWires*, ‘*filename*’);						
Script parameters:	*inpu_bef* - input image before using median filter *input_aft* - input image after using median filter *output* - output image ,a black image with white middles of visible wires *Resolution* - image resolution (µm) *WireNr* - model of number of following wires (W1, W6, W10, W13) *VisibleWire* – number of visible wires (1-7) ‘*filename*’- name of the file and directory where results will be save						

Example script:	chdir('C:\\Test Welds - Technic Control\\IQI');
	original1 = imread('iqi_test_aft.tif');
	handle1 = imshow('original1',original1,0);
	iqiImage =IQImax(original1,50,6,6,'C:\\Test Welds - Technic Control\\IQIsData.txt');
	handle2 = imshow('iqiImage',iqiImage,0);						

Remarks:	none						

LENGTHAREA

Syntax:							
Parameters:							
Description:	Detects a ROI in X direction and cut image according to EN 1435 or according to specified length.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.12.3	**Date of last update:**	2007.12.3	**Version:**	2.0
History of updates:	none						
Script syntax:	*output* = lengthArea (*input, SpecL, WeldClass, WeldThickness, SrcDist,Resolution*);						
Script parameters:	*input* - input image *output* - output image *SpecL* - specified a length [mm] by which the weld image is shorten. If value is set to 0 this length is calculated according to EN 1435 using *WeldClass, WeldThickness, ScrDist and Resolution* parameters *WeldClass* - 0 class A, 1 class B *WeldThickness* - the thickness of a weld *SrcDist* - distance between X-ray source and examined component *Resolution* - image resolution (µm)						

Example script:	chdir('C:\\');
	% read image
	a = imread('FilmFree_A_Szczecin_NA_NA_NA_TC_Spoiny_27-09-2007_14-40-04_171A.tif');
	% show image
	handle = imshow('original',a,1);
	% crop withe area
	b = cropImage(a);
	% show after crop
	handle1 = imshow('croped',b,1);
	% detect ROI according to X dimension (length of weld image)
	c = lengthArea(b,0,0,16,730,50);
	% or
	c = lengthArea(b,10,0,0,0,0);
	% show result
	handle2 = imshow('RoiX',c,1);						

Remarks:	none						

MATRIXREAD

Syntax:							
Parameters:							
Description:	Reads a matrix from textfile. This command uses Matlab method of defining matrices.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	0.9
History of updates:	none						
Script syntax:	*matrix* = **matrixread**(‘*filename*’);						
Script parameters:	*matrix – output matrix* *filename – name of the file with matrix*						
Example script:	*Example of file matrix.txt:* *[1,2;3,4]* *Matrix* = **matrixread**(‘*matrix.txt*’); *% now matrix is 1 2* *% 3 4*						
Remarks:	Use [] to concentrate numbers into matrix. Commas define columns whereas ; defines next row. This is obsolete function, use *dm* instead.						

MATRIXWRITE

Syntax:							
Parameters:							
Description:	Writes a matrix to textfile. This command uses Matlab method of defining matrices.						
References:							
List of used functions:	none						
File name (C-code):		**DLL – library name:**					
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	**matrixwrite**(*matrix*, ‘*filename*’);						
Script parameters:	*matrix – matrix to be written to disk* *filename – name of the file*						
Example script:							
Remarks:	Use [] to concentrate numbers into matrix. Commas define columns and ; defines next row.						

MEDIAN

Syntax:							
Parameters:							
Description:	2-D median filtering. The median command applies an two-dimensional median filter of size x x y to input image. Output image *output* has the same size as input.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	None						
Script syntax:	*output* = **median**(*inpu,type,versize,horsizet*);						
Script parameters:	*input* - input binary image *type* - type of mask, 0 – rectangle, 1 – cross, 2 – X *versize* - vertical size of mask *horsize* - horizontal size of mas						

Example script:	chdir('C:\\addip\radiograms');
	% read the image
	original = imread('TestSample_2_3.tif');
	% and display it on the screen (without scaling)
	handle1 = imshow('original',*original*,0);
	% median filtering
	med = **median**(*original*,0,51,51);
	% displaying result
	handle2 = imshow('ith',*med*,0);						

Remarks:	none						

MULT_ADD

Syntax:							
Parameters:							
Description:							
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**	mult_add.dll			
Developed by:	PSz	**Date of development:**	2007.09.20	**Date of last update:**	2007.09.20	**Version:**	1.0
History of updates:	None						
Script syntax:	*outimage* = **mult_add**(*inpImage*, *Mult, Add*);						
Script parameters:	*inpImage* - input original image *Mult* - number which multiplie value of each pixel *Add* - number added to each value						

Example script:	chdir('C:\\addip\\radiograms');
	or = imread('TestSample_2_3.tif');
	mm = dm('[0,0,0,0]');
	img = **mult_add**(or,10,33);
	[max,min,sred,med]=statistic(img,mm);
	imwrite(‘result.tif’,img);
	show = imshow('result', obraz,0);
	savestat('stat.txt',max,min,sred,med);						

Remarks:							

NEIGHB

Syntax:							
Parameters:							
Description:	Calculates distances between object in range specified by user.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**	neighb.dll			
Developed by:	PSz	**Date of development:**	2008.04.18	**Date of last update:**	2008.04.18	**Version:**	1.0
History of updates:							
Script syntax:	*distances* = **neighb**(*input*,*range*,*resolution*);						
Script parameters:	*input* - input image (indexed) *distances* - output matrix with distances between objects *range* - user specified range *resolution* - prescale factor (millimeters<->pixels)						

Example script:	chdir('C:\\addip\radiograms');
	% read image
	indexed = readtiff('Ind.tif');
	% distances calculation
	list = **neighb**(indexed,50,1);						

Remarks:	When the resolution parameter is set to 1, then the output matrix contain distances in pixels, otherwise in millimeters (if this value is valid). For test purposes results matrix (results.m) and distances matrix (distances.m) are saved in main directory at drive C:\						

<table>
<tr><th colspan="8">NEURAL</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Performs flaw’s detection algorithm using Artificial Neural Network (ANN) technique. The filtering window moves from pixel to pixel over an input image. Acquired data (intensities of pixels) are given as an input of previously trained nonlinear neural filter. For each position of the window one point of output image is estimated. Operates on gray scale images. Output image has the same size as input.</td></tr>
<tr><td><b>References:</b></td><td colspan="7">Shaun W. Lawson, Graham A. Parker: “<i>Intelligent segmentation of industrial radiographic images using neural networks</i>”, Proc. SPIE Vol. 2347, p. 245-255, Machine Vision Applications, Architectures, and Systems Integration III, 1994</td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7"></td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.04.12</td><td><b>Date of last update:</b></td><td>2007.04.12</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><i>output</i> = <b>neural</b>(<i>input</i>, ‘<i>filename</i>’);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>input</i> - input image
<i>output</i> - output image
‘<i>filename</i>’ - name of the file where the ANN and mask structure are stored</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">chdir('C:\\addip\radiograms');
% reading the image
<i>original</i> = imread('TestSample_2_3.tif');
% displaying it on the screen (without scaling)
<i>handle1</i> = imshow('original',<i>original</i>,0);
% defects detection by means of neural filter
<i>nn_out</i> = <b>neural</b>(<i>original</i>,'c:\\addip\\scipts\\crossMask_33x33_ANN_ff_14_7_1.ann');
% removing small dots
<i>out_med</i> = median(<i>nn_out</i>,1,5,5);
% displaying result
<i>handle2</i> = imshow('nn',<i>out_med</i>,1);</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

NORMALIZE

Syntax:							
Parameters:							
Description:	Normalize the input image to range of 0-1						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.04.12	**Date of last update:**	2007.04.12	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **normalze**(*input*);						
Script parameters:	*input* - input image *output* - output image						

Example script:	chdir('C:\\addip\radiograms');
	% reading the image
	original = imread('TestSample_2_3.tif');
	% displaying it on the screen (without scaling)
	handle1 = imshow('original',*original*,0);
	% image normalization
	norm = **normalize**(*original*);
	% displaying result
	handle2 = imshow('norm',*norm,0*);						

Remarks:	none						

NORM_RADIO

Syntax:							
Parameters:							
Description:	Special normalization						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**	norm_radio.dll			
Developed by:	PSz	**Date of development:**	2007.09.20	**Date of last update:**	2007.09.20	**Version:**	1.0
History of updates:	None						

Script syntax:	*table=dm('[x1,y1,x2,y2]');*
	outimage = **norm_radio**(*inpImage, Inum, Ibasetarget, Nmin, DIbase, table*);						

Script parameters:	*inpImage* - input original image *Inum* - number of levels *Ibasetarget* - base target *Nmin* - relative minimum *DIbase* - delta base *table* - region of interest						

Example script:	chdir('C:\\addip\radiograms');
	or = imread('TestSample_2_3.tif');
	mm = dm('[0,0,0,0]');
	img= **norm_radio**(or,65535,1000,2,8,mm);
	[max,min,sred,med]=statistic(img,mm);
	imwrite('result.tif',img);
	drukuj = imshow('result',img ,0);
	savestat('stat.txt',max,min,sred,med);						

Remarks:							

NTH

Syntax:							
Parameters:							
Description:	Global thresholding; operates on gray scale images The threshold is calculated for each pixel of an image using mean value of neighbours and standard deviation, object and background.; output is a binary image						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **nth**(*input input, MaskSize, ThCoef*);						
Script parameters:	*Input* - input image; *output* - output image *MaskSize* – size of the moving mask which contains neighborhood of selected image point *ThCoef* – threshold coefficient, that moves threshold value						

Example script:	chdir('C:\\addip\radiograms');
	% using Niblack thresholding
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = median(*a*,0,51,51);
	out_sub = imsub(*a*,*out_med*);
	out_ith = **nth**(*out_sub*,41,-0.9);
	out_med1 = median(*out_ith*,0,5,5);
	handle1 = imshow('nth',*out_med1*,1);						

Remarks:	none						

OTH

Syntax:							
Parameters:							
Description:	Global thresholding; operates on gray scale images; the calculation of threshold is based on discrimination analysis that separate statistically image on two classes, object and background.; output is a binary image						
References:							
List of used functions:	none						
File name (C-code):		**DLL – library name:**					
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **oth**(*input, ThCoef,*);						
Script parameters:	*input* - input image; *output* - output image with a 0,1 binary information where 1 is object and 0 is a background *ThCoef* - threshold coefficient, that moves threshold value						

Example script:	chdir('C:\\addip\radiograms');
	% using Otsu thresholding
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = median(*a*,0,51,51);
	out_sub = imsub(*a*,*out_med*);
	out_ith = **oth**(*out_sub*,0.4,8);
	out_med1 = median(*out_ith*,0,9,9);
	handle1 = imshow('ith',*out_med1*,1);						

Remarks:	none						

OUTLINE

Syntax:							
Parameters:							
Description:	Draws a border (1 pixel width) of any object on binary image.						
References:							
List of used functions:							
File name (C-code):		**DLL – library name:**					
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **outline**(*input*);						
Script parameters:	*input* - input image *output* - output image						

Example script:	chdir('C:\\addip\radiograms');
	% using eikiv thresholding
	a = imread('TestSample_2_3.tif');
	handle = imshow('Original',*a*,0);
	out_med = median(*a*,0,51,51);
	out_sub = imsub(*a*,*out_med*);
	out_ith = eth(*out_sub*,35,3,10,0.97,128);
	out_med1 = median(*out_ith*,0,9,9);
	out_out = **outline**(*out_med1*);
	handle1 = imshow('Outline',*out_out*,1);						

Remarks:							

PREWITT

Syntax:							
Parameters:							
Description:	Prewitt gradient filter.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **prewitt**(*input*, *type*);						
Script parameters:	*Input* - input image; *output* - output image *type* - type of mask used (0 – horizontal or 1 - vertical)						

Example script:	chdir('C:\\addip\radiograms');
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = median(*a*,0,11,11);
	out_prw = **prewitt**(*out_med,0*);
	handle1 = imshow('prewitt_0',*out_prw*,1);						

Remarks:							

READTIFF

Syntax:							
Parameters:							
Description:	Reads16-bit grayscale TIFF image.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**	readtiff.dll			
Developed by:	PSz	**Date of development:**	2007.09.26	**Date of last update:**	2007.09.26	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = **readtiff** (‘*filename*’);						
Script parameters:	*output* - image ‘*filename*’ - path to the image						

Example script:	chdir('C:\\addip\radiograms');
	
	% read image
	original = readtiff('*TestSample_3_2.tif*');
	
	% and display it on the screen (with scaling)
	handle1 = imshow(*'original',original,1*);						

Remarks:	Image read by readtiff function should be displayed with scaling. Only 16-bit images can be read using this function.						

ROBERTS

Syntax:							
Parameters:							
Description:	Roberts gradient filter.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **roberts**(*input*, *type*);						
Script parameters:	*Input* - input image; *output* - output image *type* - type of mask used (0 – $\begin{vmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{vmatrix}$ or 1 - $\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{vmatrix}$)						

Example script:	chdir('C:\\addip\radiograms');
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = median(*a*,0,11,11);
	out_rob = **roberts**(*out_med,0*);
	handle1 = imshow('roberts',*out_rob*,1);						

Remarks:							

ROTATEIMAGE

Syntax:							
Parameters:							
Description:	Rotate image clockwise by an given angle. The background is set on Background Color						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.12.3	**Date of last update:**	2007.12.3	**Version:**	2.0
History of updates:	none						
Script syntax:	*output* = rotateImage (*input, angle,BGColor*);						
Script parameters:	*input* - input image *output* - output rotated image *angle* - the angle of clockwise image rotation *BgColor* - he color of background (0 = black , 65535 = white)						

Example script:	chdir('C:\\');
	% read image
	a = readtiff('TestSample_3.2.tif');
	% show readed image
	handle = imshow('original',a,0);
	% rotate image
	out = rotateImage(a,6,65535);
	% show result
	handle1 = imshow('out',out,0);						

Remarks:	none						

<table>
<tr><th colspan="8">SAUVOLA</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Multiprocessor implementation of Sauvola thresholding.</td></tr>
<tr><td><b>References:</b></td><td colspan="7">[1] Faisal Shafait, Daniel Keysers, Thomas M. Breuel: <i>“Efficient Implementation of Local Adaptive Thresholding Techniques Using Integral Images”</i></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3">Sauvola2.dll</td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2009.03.4</td><td><b>Date of last update:</b></td><td>2009.03.4</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7">output = <b>Sauvola</b>(<i>input, w, k</i>);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>output</i> - image after thresholding
input - image to be thresholded
<i>w</i> - mask’s size (has to be odd)
<i>k</i> - k parameter of Sauvola thresholding – typically 0.2</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">%setting number of processors
threads(2);
%reading file
im = readtiff('14-35-49_257A_flaw1A_normal.tif');
%thresholding
A = Sauvola(org, 41, 0.2);
%showing image before and after thresholding
handle1 = imshow('Before', org, 1);
handle2 = imshow('After', A, 1);</td></tr>
</table>

<table>
<tr><th colspan="8">SAVEFEAT</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Saving features in to the txt file; (new version of SEAVEFEATURES)</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3">savefeat.dll</td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.03.20</td><td><b>Date of last update:</b></td><td>2007.09.26</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">None</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><b>svtxt</b><i>(tab, number, path)</i></td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>tab</i> – table with features
<i>number</i> – number of defects
<i>path</i> – path with file name</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">chdir('C:\\');
% read the image
<i>original</i> = imread('TestSample_2_3.tif');
% and display it on the screen (without scaling)
<i>handle1</i> = imshow('original',<i>original</i>,0);
% median filtering
<i>med</i> = fastmedian(<i>original</i>,51,51);
% substracting filtered image from original image
<i>sub</i> = imsub(<i>original</i>,<i>med</i>);
% iterative thresholding
<i>th</i> = ith(<i>sub</i>,0.8);
% removing small dots
<i>out_med</i> = fastmedian(<i>th</i>,5,5);
% recognizing flaws
<i>indexed_image</i> = index(<i>out_med</i>);
[<i>num</i>,<i>tab</i>] = calcfeat(<i>original</i>,<i>indexed_image</i>,<i>out_med</i>,30,100000);
<b>savefeat</b>(<i>tab</i>,<i>num</i>,'wynik.txt');
<i>im_f</i> = genindexedimg(<i>original</i>,<i>tab</i>,<i>num</i>);
% displaying result
<i>handle2</i> = imshow('ith',<i>out_med</i>,1);
<i>handle3</i> = imshow('numbered',<i>im_f</i>,0);</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">-</td></tr>
</table>

SAVESTAT

Syntax:							
Parameters:							
Description:	Saves to file statistical information						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**	savestat.dll			
Developed by:	PSz	**Date of development:**	2007.09.20	**Date of last update:**	2007.09.20	**Version:**	1.0
History of updates:	None						
Script syntax:	**savestat**(‘*path’, max*, *min,sred,med*);						
Script parameters:	*path* - path and name of file *max,min,sred,med* - data to save						

Example script:	chdir('C:\\addip\\radiograms');
	or = imread('TestSample_2_3.tif');
	mm = dm('[0,0,0,0]');
	img= norm_radio(or,65535,1000,2,8,mm);
	[max,min,sred,med]=statistic(img,mm);
	imwrite('result.tif',img);
	drukuj = imshow('result', result,0);
	savestat('stat1.txt',max,min,sred,med);						

Remarks:							

SAVEIMAGEM							
Syntax:							
Parameters:							
Description:	Saves image in the Matlab format. Under Matlab one has to use a special function (m-file) to import this file. The m-function is available in examples dir.						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:	none						
Script syntax:	**saveimagem**(‘*filename*’, *image*);						
Script parameters:	*filename* – file where image will be stored *image* – image to be saved						
Example script:	*% read the image* *original1* = imread('*h:\\nn\\TestSample_1_1.tif*'); *% saving* **saveimagem**(*‘c:\\output.bin’, original1*); *% run the following line under Matlab to import saved image* *% out = addipimport(‘c:\\output.bin’)*						
Remarks:							

SOBEL

Syntax:							
Parameters:							
Description:	Sobel gradient filter.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2007.03.20	**Date of last update:**	2007.03.20	**Version:**	1.0
History of updates:							
Script syntax:	*output* = **sobel**(*input*, *type*);						
Script parameters:	*Input* - input image; *output* - output image *type* - type of mask used (0 – horizontal or 1 - vertical)						

Example script:	chdir('C:\\addip\radiograms');
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = median(*a*,0,11,11);
	out_sob = **sobel**(*out_med,0*);
	handle1 = imshow('sobel',*out_sob*,1);						

Remarks:							

<table>
<tr><th colspan="8">STATISTIC</th></tr>
<tr><td>Syntax:</td><td colspan="7"></td></tr>
<tr><td>Parameters:</td><td colspan="7"></td></tr>
<tr><td>Description:</td><td colspan="7">Returns statistical data</td></tr>
<tr><td>References:</td><td colspan="7"></td></tr>
<tr><td>List of used functions:</td><td colspan="7">none</td></tr>
<tr><td>File name (C-code):</td><td colspan="2"></td><td colspan="2">DLL – library name:</td><td colspan="3">statistic.dll</td></tr>
<tr><td>Developed by:</td><td>PSz</td><td>Date of development:</td><td>2007.09.20</td><td>Date of last update:</td><td>2007.09.20</td><td>Version:</td><td>1.0</td></tr>
<tr><td>History of updates:</td><td colspan="7">None</td></tr>
<tr><td>Script syntax:</td><td colspan="7">*table* = dm('[*x1,y1,x2,y2*]');
[max,min,mid,med] = **statistic**(*inpImage*, *table*);</td></tr>
<tr><td>Script parameters:</td><td colspan="7">*inpImage* - input original image
table - analyze region</td></tr>
<tr><td>Example script:</td><td colspan="7">chdir('C:\\addip\\radiograms');
img = imread('TestSample_2_3.tif');
mm = dm('[0,0,0,0]');
img = norm_radio(or,65535,1000,2,8,mm);
[max,min,sred,med]=**statistic**(img,mm);
imwrite(‘result.tif',img);
drukuj = imshow('result', img,0);
savestat('stat.txt',max,min,sred,med);</td></tr>
<tr><td>Remarks:</td><td colspan="7"></td></tr>
</table>

<table>
<tr><td colspan="8"><b>STH</b></td></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Global thresholding; operates on gray scale images The threshold is calculated for each pixel of an image using mean value of neighbours and adaptive standard deviation; output is a binary image</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.03.20</td><td><b>Date of last update:</b></td><td>2007.03.20</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><i>output</i> = <b>sth</b>(<i>input input, MaskSize, ThCoef, VarDynRange)</i>;</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7">*Input* - input image;
output - output image
MaskSize – size of the moving mask which contains neighborhood of selected image point
ThCoef – threshold coefficient, that moves threshold value
VarDynRange - dynamic range of variance</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">chdir('C:\\addip\radiograms');
% using Sauvola thresholding
<i>a</i> = imread('TestSample_2_3.tif');
<i>handle</i> = imshow('original',<i>a</i>,0);
<i>out_med</i> = median(<i>a</i>,0,51,51);
<i>out_sub</i> = imsub(<i>a</i>,<i>out_med</i>);
<i>out_ith</i> = <b>sth</b>(<i>out_sub</i>,35,0.06,32768);
<i>out_med1</i> = median(<i>out_ith</i>,0,5,5);
<i>handle1</i> = imshow('sth',<i>out_med1</i>,1);</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

STHad

Syntax:							
Parameters:							
Description:	Global thresholding; operates on gray scale images The threshold is calculated for each pixel of an image using mean value of neighbours and adaptive standard deviation; output is a binary image. The dynamic range is set automatically						
References:							
List of used functions:	none						
File name (C-code):			**DLL – library name:**				
Developed by:	PSz	**Date of development:**	2008.01.04	**Date of last update:**	2008.01.04	**Version:**	1.0
History of updates:	none						
Script syntax:	*output* = STHad(*input, MaskSize, ThCoef)*;						
Script parameters:	*Input* - input image; *output* - output image *MaskSize* – size of the moving mask which contains neighborhood of selected image point *ThCoef* – threshold coefficient, that moves threshold value *VarDynRange* - dynamic range of variance						

Example script:	chdir('C:\\addip\radiograms');
	% using Sauvola thresholding
	a = imread('TestSample_2_3.tif');
	handle = imshow('original',*a*,0);
	out_med = median(*a*,0,51,51);
	out_sub = imsub(*a*,*out_med*);
	out_ith = **sth**(*out_sub*,35,0.06);
	out_med1 = median(*out_ith*,0,5,5);
	handle1 = imshow('sth',*out_med1*,1);						

Remarks:	none						

<table>
<tr><th colspan="8">THREADS</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">This function initializes global variable <i>numofproc</i>. Global variables are available for all other functions.</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7">none</td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2009.01.15</td><td><b>Date of last update:</b></td><td></td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><i>threads(numofproc);</i></td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>Numofproc</i> – number of processors to use by ADDIP (only particular functions supports multi processor systems)</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">threads(2);
chdir('C:\\');
org=readtiff('wzorzec.tif');
bw = readtiff('wzorzec.tif');
ind = index(bw);
handle3 = imshow('org',org,1);
handle4 = imshow('ind',ind,1);
writetiff(ind,'ind.tif',0);
c = cf(org,ind,bw); % now function cf uses two processors
matrixwrite(c,'features_out.m');</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">none</td></tr>
</table>

<table>
<tr><th colspan="8">WELDANGLE</th></tr>
<tr><td>**Syntax:**</td><td colspan="7"></td></tr>
<tr><td>**Parameters:**</td><td colspan="7"></td></tr>
<tr><td>**Description:**</td><td colspan="7">Function detect the angle of weld between a horizontal line and a maximum values line</td></tr>
<tr><td>**References:**</td><td colspan="7"></td></tr>
<tr><td>**List of used functions:**</td><td colspan="7"></td></tr>
<tr><td>**File name (C-code):**</td><td colspan="2"></td><td colspan="2">**DLL – library name:**</td><td colspan="3"></td></tr>
<tr><td>**Developed by:**</td><td>PSz</td><td>**Date of development:**</td><td>2007.12.3</td><td>**Date of last update:**</td><td>2007.12.3</td><td>**Version:**</td><td>2.0</td></tr>
<tr><td>**History of updates:**</td><td colspan="7">none</td></tr>
<tr><td>**Script syntax:**</td><td colspan="7">*output* = weldAngle (*input,SideWeldProfile*);</td></tr>
<tr><td>**Script parameters:**</td><td colspan="7">*input* - input image
output - angle
SideWeldProfile - the percent of Image resolution in x direction. The percent of weld profile at start and end of image are taken to set a maximum values line [%].</td></tr>
<tr><td>**Example script:**</td><td colspan="7">chdir('C:\\');
%read mage
a = imread('FilmFree_A_Szczecin_NA_NA_NA_TC_Spoiny_27-09-2007_14-40-04_171A.tif');
handle = imshow('original',a,1);
% crop image
b = cropImage(a);
handle1 = imshow('croped',b,1);
% calculate weld angle of the image and rotate by this angle
ang = weldAngle(b,10);
c = rotateImage(b,ang,65535);
handle2 = imshow('rotated',c,1);
% crop image after rotation
out = cropImage(c);
handle3 = imshow('out',out,1);</td></tr>
<tr><td>**Remarks:**</td><td colspan="7">none</td></tr>
</table>

<table>
<tr><th colspan="8">WRITETIFF</th></tr>
<tr><td><b>Syntax:</b></td><td colspan="7"></td></tr>
<tr><td><b>Parameters:</b></td><td colspan="7"></td></tr>
<tr><td><b>Description:</b></td><td colspan="7">Writes16-bit grayscale TIFF image.</td></tr>
<tr><td><b>References:</b></td><td colspan="7"></td></tr>
<tr><td><b>List of used functions:</b></td><td colspan="7"></td></tr>
<tr><td><b>File name (C-code):</b></td><td colspan="2"></td><td colspan="2"><b>DLL – library name:</b></td><td colspan="3">writetiff.dll</td></tr>
<tr><td><b>Developed by:</b></td><td>PSz</td><td><b>Date of development:</b></td><td>2007.10.04</td><td><b>Date of last update:</b></td><td>2007.10.04</td><td><b>Version:</b></td><td>1.0</td></tr>
<tr><td><b>History of updates:</b></td><td colspan="7">none</td></tr>
<tr><td><b>Script syntax:</b></td><td colspan="7"><b>writetiff</b> (image,‘<i>filename</i>’,<i>normalization</i>);</td></tr>
<tr><td><b>Script parameters:</b></td><td colspan="7"><i>input</i> - image
‘<i>filename</i>’ - new image name
<i>normalization</i> - 0 – without normalization, 1 –linear normalization to range 0-65535</td></tr>
<tr><td><b>Example script:</b></td><td colspan="7">chdir('C:\\addip\\radiograms');
% read image
original = readtiff('<i>TestSample_2_3.tif</i>');
% cut part from original
cut = cutroi(<i>original</i>,0,504,200,520);
% write new image
writetiff(<i>cut</i>,'<i>copy.tif</i>',0);
% read new image
new = readtiff('<i>copy.tif</i>');
% and display it on the screen (with scaling)
handle1 = imshow('<i>Original</i>',<i>original</i>,1);
handle2 = imshow('<i>New</i>',<i>new</i>,1);</td></tr>
<tr><td><b>Remarks:</b></td><td colspan="7">Only 16-bit images can be written using this function.</td></tr>
</table>

WSKPRECLASSIFIER							
Syntax:							
Parameters:							
Description:	Works with radiograms provided by WSK. This function makes early classification for flaws and non-flaws depending on size of the object. Preleminary results are written to the text file.						
References:							
List of used functions:							
File name (C-code):			**DLL – library name:**	wskclassifier.dll			
Developed by:	PSz	**Date of development:**	2009.09.18	**Date of last update:**	2009.09.18	**Version:**	1.0
History of updates:	none						
Script syntax:	*preclass* = **wskpreclassifier**(*image,indexed_image,thres_image,distances,features,report*);						
Script parameters:	*image* – original image *indexed_image* – indexed image *thres_image* – thresholded image *distances* – distances matrix (see example) *features* –features matrix *report* – report filename *preclass* – an BW image containing only objects classified as possible flaws						

Example script:	chdir('E:\\matlab\\ADDIP_test\\WSK');
	img = readtiff('ADDIP_20090831_133241_MD3061969_op160_4_zd9_Good.tif');
	imcut = cutroi(img,0,1024,600,825);
	out_med = fastmedian(imcut,0,31,31);
	out_sub = imsub(imcut,out_med);
	out_ith = ath(out_sub,45,1550);
	out_med1 = median(out_ith,0,5,5);
	indexed_image = index(out_med1);
	c = cf(imcut,indexed_image,out_med1);
	unused = neighb(indexed_image,40,1);
	distances = matrixread('c:\\results.m');
	preclass = wskpreclassifier(imcut,indexed_image,out_med1,distances,c,'test.txt');
	indexed_preclass = index(preclass);
	c_preclass = cf(preclass,indexed_preclass,preclass);
	out = wskclassifier(preclass,indexed_preclass,c_preclass,50,'test1.txt');
	handle1 = imshow('orginal',img,1);
	handle2 = imshow('cut',imcut,1);
	handle3 = imshow('med',out_med1,1);
	handle4 = imshow('pre_classification',preclass,1);
	handle5 = imshow('out_classifier',out,1);
	writetiff(indexed_image,'indexed.tif',1);
	writetiff(out_med1,'bw.tif',1);
	matrixwrite(c,'features_out.m');
	matrixwrite(distances,'distances.m');						

Remarks:	Due to error in neighb function, the distances matrix must be read separately from external file.						

<table>
<tr><th colspan="8">WSKCLASSIFIER</th></tr>
<tr><td>Syntax:</td><td colspan="7"></td></tr>
<tr><td>Parameters:</td><td colspan="7"></td></tr>
<tr><td>Description:</td><td colspan="7">Works with radiograms provided by WSK. This function makes final classification of flaws and assigns radiogram to class A or B</td></tr>
<tr><td>References:</td><td colspan="7"></td></tr>
<tr><td>List of used functions:</td><td colspan="7"></td></tr>
<tr><td>File name (C-code):</td><td colspan="2"></td><td colspan="2">DLL – library name:</td><td colspan="3">wskclassifier1.dll</td></tr>
<tr><td>Developed by:</td><td>PSz</td><td>Date of development:</td><td>2009.09.18</td><td>Date of last update:</td><td>2009.09.18</td><td>Version:</td><td>1.0</td></tr>
<tr><td>History of updates:</td><td colspan="7">none</td></tr>
<tr><td>Script syntax:</td><td colspan="7">out = wskclassifier(preclass,indexed_preclass,c_preclass,inch, report);</td></tr>
<tr><td>Script parameters:</td><td colspan="7">preclass – BW image from wskpreclassifier
indexed_preclass – indexed image
c_preclass – features matrix
inch – number of pixels for inch
report – report filename
out – an BW with 1inch marker indicated. Negative marker stands for the B class, dotted marker rejects the radiogram</td></tr>
<tr><td>Example script:</td><td colspan="7">chdir('E:\\matlab\\ADDIP_test\\WSK');
img = readtiff('ADDIP_20090831_133241_MD3061969_op160_4_zd9_Good.tif');
imcut = cutroi(img,0,1024,600,825);
out_med = fastmedian(imcut,0,31,31);
out_sub = imsub(imcut,out_med);
out_ith = ath(out_sub,45,1550);
out_med1 = median(out_ith,0,5,5);
indexed_image = index(out_med1);
c = cf(imcut,indexed_image,out_med1);
unused = neighb(indexed_image,40,1);
distances = matrixread('c:\\results.m');
preclass = wskpreclassifier(imcut,indexed_image,out_med1,distances,c,'test.txt');
indexed_preclass = index(preclass);
c_preclass = cf(preclass,indexed_preclass,preclass);
out = wskclassifier(preclass,indexed_preclass,c_preclass,50,'test1.txt');
handle1 = imshow('orginal',img,1);
handle2 = imshow('cut',imcut,1);
handle3 = imshow('med',out_med1,1);
handle4 = imshow('pre_classification',preclass,1);
handle5 = imshow('out_classifier',out,1);
writetiff(indexed_image,'indexed.tif',1);
writetiff(out_med1,'bw.tif',1);
matrixwrite(c,'features_out.m');
matrixwrite(distances,'distances.m');</td></tr>
<tr><td>Remarks:</td><td colspan="7">Due to error in neighb function, the distances matrix must be read separately from external file.</td></tr>
</table>