

Refresh

- ☐ Show parent functions
- ☒ Show busy lines
- ☒ Show child functions
- ☐ Show M-Lint results
- ☒ Show file coverage
- ☒ Show function listing

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
127	outImg = exp(idwt(hpChannels, ...	1	0.359 s	54.7%	<div></div>
98	[hpChannels lpChannels] = dwt(...	1	0.063 s	9.6%	<div></div>
38	addpath common\;	1	0.063 s	9.6%	<div></div>
40	addpath artikel1\;	1	0.047 s	7.2%	<div></div>
39	addpath common\mxSimpleDiff;	1	0.031 s	4.7%	<div></div>
Other lines & overhead			0.093 s	14.2%	<div></div>
Totals			0.656 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
multiScaleEnh>idwt	M-subfunction	1	0.344 s	52.4%	<div></div>
addpath	M-function	3	0.141 s	21.5%	<div></div>
multiScaleEnh>dwt	M-subfunction	1	0.063 s	9.6%	<div></div>
multiScaleEnh>softThresh	M-subfunction	4	0.031 s	4.7%	<div></div>
multiScaleEnh>egag	M-subfunction	1	0.016 s	2.4%	<div></div>
mean	M-function	1	0.015 s	2.3%	<div></div>

IsScalar	M-function	12	0 s	0%	
im2single	M-function	1	0 s	0%	
multiScaleEnh>lookUp	M-subfunction	6	0 s	0%	
Self time (built-ins, overhead, etc.)			0.046 s	7.0%	■
Totals			0.656 s	100%	

Coverage results

[[Show coverage for parent directory](#)]

Total lines in file	364
Non-code lines (comments, blank lines)	215
Code lines (lines that can run)	149
Code lines that did run	33
Code lines that did not run	116
Coverage (did run/can run)	22.15 %

Function listing

Color highlight code according to

time

```
time    calls    line
1 function [ outImg execTime ] = multiScaleEnh(inImg,j,
2                                     sigma,t1,
3                                     sampl)
4
5 %MULTISCALEENH Multi-scale enhancement of medical ult
6 %   [ENHANCEDIMAGE EXECUTIONTIME] =
7 %   MULTISCALENH(INIMG,J,ALPHA,TMIN,TMAX,SIGMA,T1,T2,
8 %
9 %   Returns:
10 %   outImg      Processed Image.
11 %   execTime    Execution time for image enhancement
12 %
13 %   Parameters:
14 %   INIMG       -   Input image.
15 %   J           -   Number of channels to be computed by
16 %   ALPHA       -   Decreasing factor for soft threshold
17 %                  channels in DWT.
18 %   TMIN        -   Minimum scaling factor for soft thre
19 %   TMAX        -   Maximum scaling factor for soft thre
20 %   SIGMA       -   Estimate of the standard deviation c
21 %   T1          -   Hard threshold.
22 %   T2          -   Lower bound for edge enhancement thr
23 %                  Adaptive Gain of DWT coefficients.
24 %   T3          -   Upper bound for edge enhancement thr
25 %                  Adaptive Gain of DWT coefficients.
26 %   C           -   Gain in GAG-function.
27 %   B           -   Controls shape and sign of GAG-funct
```

```

28 %   GRAPHS       -   Controls wether graphs of thresholdi
29 %                   functions should be plotted or not.
30 %
31 %   Licensed under BSD as a part of EDGY project sour
32 %   see readme.txt file. For more information see pro
33 %
34 %   Revision: 1.0
35 %   Date: 2007/05/08
36 %   Author: Alexander Tuttle, Erik Ringaby
37
0.06 1 38 addpat common\;
0.03 1 39 addpat common\mxSimpleDiff;
0.05 1 40 addpat artikell\;
41
1 42 if ~(IsScalar(j) && IsScalar(alpha) && IsScalar(tMax)
43     && IsScalar(t1) && IsScalar(t2) && IsScalar(t
44     && IsScalar(b) && IsScalar(sigma) && IsScalar
45     && IsScalar(sampl))
46     error('Parameters must be scalar.');
```

47 end

```

1 48 if ~(numel(j) == 1 && isnumeric(j) &&...
49     (j > 0) && (round(j) == j) && j < 6 );
50     error('The number of levels must be a positive in
51 end
1 52 if (tMin >= tMax)
53     error('t_min must be less than t_max');
```

54 end

```

1 55 if ~(t1 >= 0 && t2 >= t1 && t3 > t2 && t3 <= 1)
56     error('t1, t2 and t3 must satisfy 0 <= t1 <= t2 <
57 end
58
59
60 % Convert the image to gray scale.
0.02 1 61 inImg = im2single(mear(inImg,3));
62
1 63 if sampl && min(size(inImg)) <= 2^j
64     error('The input image is too small to be downsam
65 end
66
67 % If sigma is specified as 0 the user is prompted to
68 % image that is adequate for estimating the standard
69 % noise.
0.02 1 70 logImg = log(inImg+eps);
1 71 if sigma == 0
72     [upperLeft lowerRight] = getUserCoordinates(inImg
73     rows = min(upperLeft(1,2),lowerRight(1,2)):max(lc
74     cols = min(upperLeft(1,1),lowerRight(1,1)):max(lc
75     imgSelection = logImg(rows,cols);
76     sigma = std(imgSelection(:));
77 end
78
79 % Compute the function table for the enhancement func
< 0.01 1 80 res = 0.001; %table resolution
< 0.01 1 81 s = -1; %lower bound for table
< 0.01 1 82 e = 1; %upper bound for table
0.02 1 83 egagTable = egac(s:res:e, b, c, t1, t2, t3);
84
```

```

85 % Plot graphs of thresholding and enhancement functionio
1 86 if graphs
87     showGraphs(j,alpha,tMin,tMax,sigma,egagTable,res)
88 end
89
90 % Start timer for measuring execution time of image e
< 0.01 1 91 tic;
92
93 % Compute the natural logarithm of the image to separ
94 % noise.
0.02 1 95 inImg = log(inImg+eps);
96
97 % ----- DWT -----
0.06 1 98 [hpChannels lpChannels] = dwt(inImg,j,sampl);
99 % -----
100
101 % ----- Soft thresholding -----
102 % The coefficients below describe the relations betwe
103 % deviation of the noise in the image and the respect
< 0.01 1 104 sigmaxCoeff = [0.7122 0.2828 0.1907 0.1472 0.1217];
< 0.01 1 105 sigmayCoeff = [0.3416 0.1946 0.1468 0.1194 0.1012];
106
107 for level = 1:floor(j/2)
0.02 2 108     hpChannels{level}(:, :, 1) = softThresl(hpChannels{1
109         tMax, tMin, sigmaxCoeff(level)*sigma, alpha,
0.02 2 110     hpChannels{level}(:, :, 2) = softThresl(hpChannels{1
111         tMax, tMin, sigmayCoeff(level)*sigma, alpha,
2 112 end
113 % -----
114
115 % ----- Enhancement through Generalized Adaptiv
1 116 for iChannel = ceil(j/2):j
3 117     M1 = max(abs(reshape(hpChannels{iChannel}(:, :, 1),
3 118     M2 = max(abs(reshape(hpChannels{iChannel}(:, :, 2),
3 119     hpChannels{iChannel}(:, :, 1) = M1*lookUp(egagTable
120         hpChannels{iChannel}(:, :, 1)/M1);
3 121     hpChannels{iChannel}(:, :, 2) = M2*lookUp(egagTable
122         hpChannels{iChannel}(:, :, 2)/M2);
3 123 end
124 % -----
125
126 % ----- IDWT -----
0.36 1 127 outImg = exp(idwt(hpChannels, lpChannels, sampl));
128 % -----
< 0.01 1 129 execTime = toc;
130
131 % ##### Help functions #####
132
133 % ----- DWT -----
134 function [hpChannels lpChannel] = dwt(img,numChannels
135 %DWT Frequency decomposition of 2D signal.
136 % [HP LP] = DWT(S,N,D) decomposes the signal into f
137 % according to the follow illustration:
138 %
139 %     |--HPx{1}
140 %     |
141 %     |--HPy{1}           |--HPx{N}

```

```

142 % | |
143 % S--- | |--HPy{N}
144 % | |
145 % |--LP--- ... ---|
146 % |
147 % |--LP
148 %
149 % S - Signal to be decomposed.
150 % N - Number of channels to compute.
151 % D - Downsample LP-component between channels.
152
153 % Filter coefficients for analyzing filters.
154 hx = [0.0625 0.25 0.375 0.25 0.0625];
155 gx = [0 1 -1];
156
157 lpChannel = img;
158
159 if(sampl)
160     for iChannel = 1:numChannels
161         hpChannels{iChannel}(:, :, 1) = mxSimpleDiff(lp
162             hpChannels{iChannel}(:, :, 2) = mxSimpleDiff(lp
163
164         lpChannel = imfilter(imfilter(lpChannel, hx, 'r
165             hx', 'replicate', 'conv'));
166         lpChannel = lpChannel(1:2:end, 1:2:end);
167     end
168 else
169     for iChannel = 1:numChannels
170         hpChannels{iChannel}(:, :, 1) = mxSimpleDiff(lp
171             2^(iChannel-1)-1);
172         hpChannels{iChannel}(:, :, 2) = mxSimpleDiff(lp
173             2^(iChannel-1)-1)';
174
175         lpChannel = imfilter(imfilter(lpChannel, zeroP
176             'replicate', 'conv'), zeroPad(hx, iChannel)
177     end
178 end
179 % -----
180
181 % ----- IDWT -----
182 function img = idwt(hpChannels, lpChannel, sampl)
183 %IDWT Reconstructs a signal that has been decomposed
184 % S = IDWT(HPCHANNELS, LPCHANNEL, SAMPL)
185 % HPCHANNELS, LPCHANNEL - Output from DWT, write 'h
186 % SAMPL - Boolean that indicates we
187 % downsampled during decomp
188
189 % Filter coefficients for reconstructing filters.
190 hx = [0.0625 0.25 0.375 0.25 0.0625];
191 kx = [-0.00390625 -0.03515625 -0.14453125 -0.36328125
192     0.14453125 0.03515625 0.00390625 0];
193 lx = [0.001953125 0.015625 0.0546875 0.109375 0.63671
194     0.0546875 0.015625 0.001953125];
195
196 % Split signal into channels with or without downsamp
197 % channels.
198 iChannel = length(hpChannels);

```

```

199 if(sampl)
200     while iChannel > 0
201         hpChannelX = imfilter(imfilter(hpChannels{iCh
202             lx', 'replicate', 'conv'), kx, 'replicate'
203
204         hpChannelY = imfilter(imfilter(hpChannels{iCh
205             kx', 'replicate', 'conv'),lx, 'replicate'
206
207         lpChannel = interp2(lpChannel,'spline');
208         if mod(size(hpChannelX,1),2) == 0
209             lpChannel = [lpChannel;lpChannel(end,:)];
210         end
211         if mod(size(hpChannelX,2),2) == 0
212             lpChannel = [lpChannel lpChannel(:,end)];
213         end
214
215         lpChannel = imfilter(imfilter(lpChannel,hx','
216             'replicate');
217         lpChannel = hpChannelX + hpChannelY +lpChanne
218         iChannel = iChannel-1;
219     end
220 else
221     while iChannel > 0
222         hpChannelX = imfilter(imfilter(hpChannels{iCh
223             zeroPad(lx,iChannel)', 'replicate', 'conv
224             zeroPad(kx,iChannel), 'replicate', 'conv'
225
226         hpChannelY = imfilter(imfilter(hpChannels{iCh
227             zeroPad(kx,iChannel)', 'replicate', 'conv
228             zeroPad(lx,iChannel), 'replicate', 'conv'
229
230         lpChannel = imfilter(imfilter(lpChannel,zeroP
231             'replicate'),zeroPad(hx,iChannel),'repl
232
233         lpChannel = hpChannelX + hpChannelY +lpChanne
234         iChannel = iChannel-1;
235     end
236 end
237 img = lpChannel;
238 % -----
239
240 % ----- SOFTTHRESH -----
241 function U = softThresh( V, tMax, tMin, sigma, alpha,
242 %SOFTTHRESH performs soft thresholding of the values
243 % Y = softThresh( X, TMAX, TMIN, S, A, L ) performs
244 % thresholding of X with the threshold t computed a
245 %
246 % ( (TMAX-A*(L-1))*S if TMAX-A*(L-1)>TMIN
247 % t = <
248 % ( TMIN*S otherwise
249
250 % Calculate a threshold
251 c = tMax - alpha*(lev-1);
252
253 if c > tMin
254     t = c*sigma;
255 else

```

```

256         t = tMin*sigma;
257     end
258 % Apply soft thresholding
259 U = sign(V).*(abs(V)-t).*(abs(V) > t);
260 % -----
261
262 % ----- EGAG -----
263 function Y = egag( X, b, c, t1, t2, t3 )
264 %EGAG Signal enhancement of X through Generalized Ada
265 %   Y = EGAG( X, B, C, T1, T2, T3 ).
266 %   X is the signal that you want to enhance, it can
267 %   a matrix. Parameters B and C control the amount o
268 %   T2 and T3 are thresholds. Values in [0 T1] are ma
269 %   [T2 T3] are amplified and values in ]T1 T2[ and a
270 %   altered.
271
272 a = 1./(sigm(c*(1-b))-sigm(-c*(1+b)));
273 signx = sign(X);
274 U = signx.*(abs(X)-t2)./(t3-t2);
275 U_bar = a*(t3-t2)*(sigm(c*(U-b))-sigm(-c*(U+b)));
276
277 Y = (signx.*t2+U_bar).*((abs(X) <= t3).*((abs(X) >= t
278     X.*((abs(X) > t3) + (abs(X) < t2)).*(abs(X)>t1);
279
280 function Y = sigm( X )
281 %SIGM A sigmoid function
282 %   Y = SIGM(X) = 1./(1+exp(-X));
283
284 Y = 1./(1+exp(-X));
285 % -----
286
287 % ----- ZEROPAD -----
288 function out = zeroPad(in, n)
289 %ZEROPAD Insert zeros between elements of a vector or
290 %   Y = ZEROPAD(X,N) inserts 2^(N-1)-1 zeros between
291 %   vector or matrix X. If X is a matrix X will be ze
292 %   dimensions.
293
294 if (n <= 0)
295     error('n must be equal to or larger than one');
296 end
297 if (n == 1)
298     out = in;
299 else
300     n = 2^(n-1);
301     out = zeros(size(in,1),n*size(in,2)-n+1);
302     out(:,1:n:end)=in;
303 end
304 % -----
305
306 % ----- LOOKUP -----
307 function Y = lookUp(funTable,res,X)
308 %LOOKUP Look up values in a function table with NN-in
309 %   Y = LOOKUP(FUNTABLE, R, X)
310 %   FUNTABLE - table of function values.
311 %   R         - table resolution.
312 %   X         - values whos correspodng function valu

```

```

313 % are to be looked up.
314
315 % Dummy variables intended to optimize speed
316 tVar1 = (1/res); % Multiplication is faster than divi
317 tVar2 = length(funTable)/2;
318
319 Y = funTable(ceil(X*tVar1+tVar2));
320 % -----
321
322 % ----- SHOWGRAPHS -----
323 function showGraphs(j,alpha,tMin,tMax,sigma,egagTable
324 %SHOWGRAPHS Displays graphs of thresholding and enhan
325 x = linspace(-1,1,1/res);
326 numC = floor(j/2);
327 % Plot softThresh for fine scale levels of DWT
328 fig1 = figure;
329 set(fig1,'Name','Thresholding and enhancement functio
330 for channel = 1:numC
331     subplot(1,numC+1,channel);
332     plot(x,softThresh(x, tMax, tMin, sigma, alpha, ch
333     axis square;title(sprintf('softThreshold in chann
334     xlabel x;ylabel('softThresh(x)');
335 end
336 %Plot egag function
337 subplot(1,numC+1,numC+1);plot(x,lookUp(egagTable,res,
338 title('Generalized Adaptive Gain function');xlabel x;
339 % -----
340
341 function [ x1 x2 ] = getUserCoordinates(img)
342 %GETUSERCOORDINATES
343 % [ X1 X2 ] = GETUSERCOORDINATES(IMG)
344 % Lets the user choose two points in an image and ret
345 % the coordinates of each point in the row vectors X1
346
347 x1 = zeros(1,2);
348 x2 = zeros(1,2);
349
350 fig = figure;
351 imshow(img,[min(img(:)),max(img(:))]);axis image;colo
352 set(fig,'Name','Select area for estimation of standar
353
354 disp('Select upper left corner of preferred area with
355 t = waitforbuttonpress;
356 [x1(1,1) x1(1,2)] = ginput(1);
357 disp('Select lower right corner of preferred area wit
358 t = waitforbuttonpress;
359 [x2(1,1) x2(1,2)] = ginput(1);
360
361 x1 = ceil(x1);
362 x2 = floor(x2);
363 close(fig);
364 % -----

```