

multiScaleEnh (1 call, 0.733 sec)

Generated 10-May-2007 01:36:47 using real time.

M-function in file [H:\edgy\Implementering\profilekoder\artikel2\multiScaleEnh.m](#)
[\[Copy to new window for comparing multiple runs\]](#)

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	0.250 s	34.1%	<div></div>
98	[hpChannels lpChannels] = dwt(...	1	0.109 s	14.9%	<div></div>
38	addpath common\;	1	0.062 s	8.5%	<div></div>
40	addpath artikel1\;	1	0.046 s	6.3%	<div></div>
119	hpChannels{iChannel}(:, :, 1) = ...	3	0.032 s	4.4%	<div></div>
Other lines & overhead			0.234 s	31.9%	<div></div>
Totals			0.733 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
multiScaleEnh>idwt	M-subfunction	1	0.234 s	31.9%	<div></div>
addpath	M-function	3	0.140 s	19.1%	<div></div>
multiScaleEnh>dwt	M-subfunction	1	0.109 s	14.9%	<div></div>
multiScaleEnh>softThresh	M-subfunction	4	0.046 s	6.3%	<div></div>
multiScaleEnh>lookUp	M-subfunction	6	0.032 s	4.4%	<div></div>
mean	M-function	1	0.016 s	2.2%	<div></div>

IsScalar	M-function	12	0 s	0%	
im2single	M-function	1	0 s	0%	
multiScaleEnh>egag	M-subfunction	1	0 s	0%	
Self time (built-ins, overhead, etc.)			0.156 s	21.3%	<div></div>
Totals			0.733 s	100%	

Coverage results

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

Total lines in file	368
Non-code lines (comments, blank lines)	219
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 whether 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
1 81 s = -1; %lower bound for table
< 0.01 1 82 e = 1; %upper bound for table
1 83 egagTable = egag(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.
0.03 1 95 inImg = log(inImg+eps);
96
97 % ----- DWT -----
0.11 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)
0.03 2 108     hpChannels{level}(:, :, 1) = softThresl(hpChannels{1
109         tMax, tMin, sigmaxCoeff(level)*sigma, alpha,
0.03 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
0.03 3 117     M1 = max(abs(reshape(hpChannels{iChannel}(:, :, 1),
0.02 3 118     M2 = max(abs(reshape(hpChannels{iChannel}(:, :, 2),
0.03 3 119     hpChannels{iChannel}(:, :, 1) = M1*lookUp(egagTable,
120         hpChannels{iChannel}(:, :, 1)/M1);
0.03 3 121     hpChannels{iChannel}(:, :, 2) = M2*lookUp(egagTable,
122         hpChannels{iChannel}(:, :, 2)/M2);
3 123 end
124 % -----
125
126 % ----- IDWT -----
0.25 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 %         hpChannels{iChannel}(:, :, 1) = imfilter(lpCh
164 %         hpChannels{iChannel}(:, :, 2) = imfilter(lpCh
165
166         lpChannel = imfilter(imfilter(lpChannel, hx, 'r
167         hx', 'replicate', 'conv'));
168         lpChannel = lpChannel(1:2:end, 1:2:end);
169     end
170 else
171     for iChannel = 1:numChannels
172         hpChannels{iChannel}(:, :, 1) = mxSimpleDiff(lp
173         2^(iChannel-1)-1);
174         hpChannels{iChannel}(:, :, 2) = mxSimpleDiff(lp
175         2^(iChannel-1)-1)';
176 %         hpChannels{iChannel}(:, :, 1) = imfilter(lpCh
177 %         hpChannels{iChannel}(:, :, 2) = imfilter(lpCh
178
179         lpChannel = imfilter(imfilter(lpChannel, zeroP
180         'replicate', 'conv'), zeroPad(hx, iChannel)
181     end
182 end
183 % -----
184
185 % ----- IDWT -----
186 function img = idwt(hpChannels, lpChannel, sampl)
187 %IDWT Reconstructs a signal that has been decomposed
188 %      S = IDWT(HPCHANNELS, LPCHANNEL, SAMPL)
189 %      HPCHANNELS, LPCHANNEL - Output from DWT, write 'h
190 %      SAMPL                  - Boolean that indicates we
191 %                              downsampling during decomp
192
193 % Filter coefficients for reconstructing filters.
194 hx = [0.0625 0.25 0.375 0.25 0.0625];
195 kx = [-0.00390625 -0.03515625 -0.14453125 -0.36328125
196       0.14453125 0.03515625 0.00390625 0];
197 lx = [0.001953125 0.015625 0.0546875 0.109375 0.63671
198       0.0546875 0.015625 0.001953125];

```

```

199
200 % Split signal into channels with or without downsamp
201 % channels.
202 iChannel = length(hpChannels);
203 if(sampl)
204     while iChannel > 0
205         hpChannelX = imfilter(imfilter(hpChannels{iCh
206             lx', 'replicate', 'conv'), kx, 'replicate
207
208         hpChannelY = imfilter(imfilter(hpChannels{iCh
209             kx', 'replicate', 'conv'),lx, 'replicate'
210
211         lpChannel = interp2(lpChannel,'spline');
212         if mod(size(hpChannelX,1),2) == 0
213             lpChannel = [lpChannel;lpChannel(end,:)];
214         end
215         if mod(size(hpChannelX,2),2) == 0
216             lpChannel = [lpChannel lpChannel(:,end)];
217         end
218
219         lpChannel = imfilter(imfilter(lpChannel,hx','
220             'replicate');
221         lpChannel = hpChannelX + hpChannelY +lpChanne
222         iChannel = iChannel-1;
223     end
224 else
225     while iChannel > 0
226         hpChannelX = imfilter(imfilter(hpChannels{iCh
227             zeroPad(lx,iChannel)', 'replicate', 'conv
228             zeroPad(kx,iChannel), 'replicate', 'conv'
229
230         hpChannelY = imfilter(imfilter(hpChannels{iCh
231             zeroPad(kx,iChannel)', 'replicate', 'conv
232             zeroPad(lx,iChannel), 'replicate', 'conv'
233
234         lpChannel = imfilter(imfilter(lpChannel,zeroP
235             'replicate'),zeroPad(hx,iChannel),'reptic
236
237         lpChannel = hpChannelX + hpChannelY +lpChanne
238         iChannel = iChannel-1;
239     end
240 end
241 img = lpChannel;
242 % -----
243
244 % ----- SOFTTHRESH -----
245 function U = softThresh( V, tMax, tMin, sigma, alpha,
246 %SOFTTHRESH performs soft thresholding of the values
247 %   Y = softThresh( X, TMAX, TMIN, S, A, L ) performs
248 %   thresholding of X with the threshold t computed a
249 %
250 %       ( (TMAX-A*(L-1))*S  if   TMAX-A*(L-1)>TMIN
251 %   t = <
252 %       ( TMIN*S              otherwise
253
254 % Calculate a threshold
255 c = tMax - alpha*(lev-1);

```

```

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

```

```

313 %      Y = LOOKUP(FUNTABLE, R, X)
314 %      FUNTABLE - table of function values.
315 %      R          - table resolution.
316 %      X          - values whos correspodng function valu
317 %                  are to be looked up.
318
319 % Dummy variables intended to optimize speed
320 tVar1 = (1/res); % Multiplication is faster than divi
321 tVar2 = length(funTable)/2;
322
323 Y = funTable(ceil(X*tVar1+tVar2));
324 % -----
325
326 % ----- SHOWGRAPHS -----
327 function showGraphs(j,alpha,tMin,tMax,sigma,egagTable
328 %SHOWGRAPHS Displays graphs of thresholding and enhan
329 x = linspace(-1,1,1/res);
330 numC = floor(j/2);
331 % Plot softThresh for fine scale levels of DWT
332 fig1 = figure;
333 set(fig1,'Name','Thresholding and enhancement functio
334 for channel = 1:numC
335     subplot(1,numC+1,channel);
336     plot(x,softThresh(x, tMax, tMin, sigma, alpha, ch
337     axis square;title(sprintf('softThreshold in chann
338     xlabel x;ylabel('softThresh(x)');
339 end
340 %Plot egag function
341 subplot(1,numC+1,numC+1);plot(x,lookUp(egagTable,res,
342 title('Generalized Adaptive Gain function');xlabel x;
343 % -----
344
345 function [ x1 x2 ] = getUserCoordinates(img)
346 %GETUSERCOORDINATES
347 %      [ X1 X2 ] = GETUSERCOORDINATES(IMG)
348 % Lets the user choose two points in an image and ret
349 % the coordinates of each point in the row vectors X1
350
351 x1 = zeros(1,2);
352 x2 = zeros(1,2);
353
354 fig = figure;
355 imshow(img,[min(img(:)),max(img(:))]);axis image;colo
356 set(fig,'Name','Select area for estimation of standar
357
358 disp('Select upper left corner of preferred area with
359 t = waitforbuttonpress;
360 [x1(1,1) x1(1,2)] = ginput(1);
361 disp('Select lower right corner of preferred area wit
362 t = waitforbuttonpress;
363 [x2(1,1) x2(1,2)] = ginput(1);
364
365 x1 = ceil(x1);
366 x2 = floor(x2);
367 close(fig);
368 % -----

```