

AWMF (1 call, 8.312 sec)

Generated 09-May-2007 23:14:34 using real time.

M-function in file [H:\edgy\Implementering\profilekoder\artikel3\AWMF.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
<a href="#">116</a>	[sortInPicLocal index] = sort(...	126024	3.097 s	37.3%	<div></div>
<a href="#">112</a>	wLocal = round(max(wLocal,0));...	126024	1.016 s	12.2%	<div></div>
<a href="#">113</a>	sumW = sum(wLocal(:));	126024	0.933 s	11.2%	<div></div>
<a href="#">108</a>	inPicLocal = inPicExt(iRow-K :...	126024	0.734 s	8.8%	<div></div>
<a href="#">111</a>	wLocal = (wCenter - distMap*qu...	126024	0.578 s	7.0%	<div></div>
Other lines & overhead			1.954 s	23.5%	<div></div>
Totals			8.312 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
<a href="#">imfilter</a>	M-function	2	0.062 s	0.7%	<div></div>
<a href="#">repmat</a>	M-function	8	0.015 s	0.2%	
<a href="#">meshgrid</a>	M-function	1	0 s	0%	
Self time (built-ins, overhead, etc.)			8.235 s	99.1%	<div></div>
Totals			8.312 s	100%	

Coverage results

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

Total lines in file	157
Non-code lines (comments, blank lines)	86
Code lines (lines that can run)	71
Code lines that did run	47
Code lines that did not run	24
Coverage (did run/can run)	66.20 %

Function listing

Color highlight code according to

```
time  calls  line
1 function [outPic time] = AWMF(inPic, c, K, wCenter, g
2 %AWMF Adaptive Weighted Median Filter
3 % [OUTPIC TIME] = AWMF(INPIC, C, K, WCENTER, GRAPHS
4 % use an Adaptive Weighted Median Filter for reduci
5 % result, OUTPIC, has the same size and class as IN
6 % for the function is returned in the variable TIME
7 %
8 % The weights, [W], for a sequence, [X], extends th
9 % For example, if W = [w1=2 w2=0 w3=3] the weighted
10 % sequence x = [x1 x2 x3] is [x1 x1 x3 x3 x3].
11 % The weights for a local neighbourhood is given by
12 %
13 % W(i,j) = [WCENTER - distance * C * LocalMean / Lo
14 %
15 % For more information and details see project docu
16 %
17 % INPIC      Input grayscale picture, UINT8
18 % C          Scalar constant, C >= 0 (if 0, me
19 % K          Size of 2D filter kernel (N*N), N
20 %           K <= min(size(INPIC))
21 % WCENTER    Kernel Centrum Weight, WCENTER >=
22 % GRAPHS     if 1 shows extra graphs for devel
23 %
24 % Class support for inputs INPIC
25 % uint8
26 % Class support for inputs C, K, WCENTER, GRAPHS:
27 % double
28 %
29 % Class for output OUTPIC
30 % uint8
31 % Class for output TIME
32 % double
33 %
34 % Requirements: IMFILTER in Image Processing Toolbo
35 %
36 % See also MEDIAN.
37 %
38 % Licensed under BSD as a part of EDGY project sour
```

```

39 % see readme.txt file. For more information see pro
40 %
41 % Revision: 1.0
42 % Date: 2007/05/08
43 % Author: [Per Thyr]
44
1 45 if (~isnumeric(c) || numel(c) ~=1 || c < 0)
46     error('C must be a positive scalar.');
```

```

1 47 elseif (~isnumeric(K) || numel(K) ~=1 ||...
48     K < 0 || round(K) ~= K)
49     error('K must be a positive integer scalar.');
```

```

1 50 elseif ndims(inPic) > 2
51     error('Input picture must be in 2 dimensions')
1 52 elseif (K > min(size(inPic)))
53     error('K <= min(size(INPIC)).');
```

```

1 54 elseif (~isnumeric(wCenter) || numel(wCenter) ~=1 ||
55     error('WCENTER must be a positive scalar.');
```

```

1 56 elseif (~isnumeric(graphs) || numel(graphs) ~=1 ||...
57     graphs > 2 || graphs < 0)
58     error('GRAPHS must be 0 or 1.');
```

```

1 59 elseif (~isa(inPic, 'uint8'))
60     error('Input picture must be UINT8-class.')
61 end
62
63 % Starts calculating runningtime
```

```

< 0.01 1 64 tic
65
0.02 1 66 % Convert uint8 format to double. Add 1 to avoid div
67 inPic = double(inPic) + 1;
68
69 % Local neighbourhood sidelength and area
```

```

< 0.01 1 70 N = 2*K+1;
< 0.01 1 71 sqrN = N^2;
72
73 % Calculate quota (c*var/mean)
```

```

0.05 1 74 kernelMean = ones(N,N)/(N*N);
0.02 1 75 inPicMean = imfilter(inPic, kernelMean, 'conv', 'repl
1 76 inPicMeanSq = imfilter(inPic.^2, kernelMean, 'conv',
77
0.02 1 78 quotaWeight = c*(inPicMeanSq./inPicMean - inPicMean);
1 79 clear inPicMean inPicMeanSq;
80
81 % Makes extended quotaWight
```

```

0.02 1 82 tquotaWeightExt = [repmat(quotaWeight(:,1),1,K) quotaW
83     repmat(quotaWeight(:,end),1,K)];
1 84 quotaWeightExt = [repmat(tquotaWeightExt(1,:),K,1); t
85     repmat(tquotaWeightExt(end,:),K,1)
1 86 clear tquotaWeightExt;
87
88 % Makes extended inPic
```

```

1 89 tmpInPicExt = [repmat(inPic(:,1),1,K) inPic ...
90     repmat(inPic(:,end),1,K)];
1 91 inPicExt = [repmat(tmpInPicExt(1,:),K,1); tmpInPicExt
92     repmat(tmpInPicExt(end,:),K,1)];
0.02 1 93 clear tmpInPicExt;
94
95 % Calculate distancematrix
```

```

1 96 [dMeshCol, dMeshRow] = meshgrid(-K:K,-K:K);
1 97 distMap = sqrt((dMeshRow.^2) + (dMeshCol.^2));
98
99 % Initiation
1 100 outPic = zeros(size(inPic));
< 0.01 1 101 [numRowExt, numColExt] = size(inPicExt);
102
103 % Calculate new pixelevalue, pixel by pixel
1 104 for iRow = 1+K : numRowExt-K
177 105     for iCol = 1+K : numColExt - K
106
107         % Local Neighbourhood
0.73 126024 108         inPicLocal = inPicExt(iRow-K : iRow+K, iCol-K
109
110         % Calculate local weightmatrix, wLocal
0.58 126024 111         wLocal = (wCenter - distMap*quotaWeightExt(iR
1.02 126024 112         wLocal = round(max(wLocal,0)); % negative 2 z
0.93 126024 113         sumW = sum(wLocal(:));
114
115         % Sort
3.10 126024 116         [sortInPicLocal index] = sort(inPicLocal(:),
117
118         % Calculate median (output value)
< 0.01 126024 119         newPxPos = (sumW+1)/2;
< 0.01 126024 120         sumW = 0;
0.11 126024 121         for iPos = 1:sqrN
0.03 5169838 122             sumW = sumW + wLocal(index(iPos));
0.18 5169838 123             if sumW >= newPxPos
< 0.01 126024 124                 break
125
0.19 5043814 126             end
< 0.01 126024 127             yWM = sortInPicLocal(iPos);
< 0.01 126024 128             outPic(iRow - K, iCol - K) = yWM;
0.34 126024 129         end
177 130     end
131
132 % Development graphs
1 133 if graphs == 1
134     quota = quotaWeight./c;
135     figure('Name',['AWMF: Development graphs, c = ' n
136         ', K = ' num2str(K) ...
137         ' ( ' num2str(N) '*' num2str(N) ' )'...'
138         ', wCenter = ' num2str(wCenter) ],'NumberTitl
139     subplot(3,2,1),imagesc(inPic);
140         title('InPic'); colormap gray; axis image; co
141     subplot(3,2,2),imagesc(outPic, [min(inPic(:)) max
142         title('outPic');colormap gray; axis image; co
143     subplot(3,2,3),imagesc(quota);
144         title('variance/mean quota');colormap gray; a
145     subplot(3,2,4),imagesc(quotaWeight);
146         title('c*variance/mean quota');colormap gray;
147     subplot(3,2,5),imagesc(abs(inPic - outPic));
148         title('|inPic - outPic|'); colormap gray; axi
149     subplot(3,2,6),imagesc(distMap);
150         title('distMap'); colormap gray; axis image;
151     end
152

```

```
153 % Convert back to uint8
1 154 outPic = uint8(outPic - 1);
155
156 % Stops calculating runnningtime
1 157 time = toc;
```

< 0.01