

```

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

```