ROBUST CONTEXT DENOISING OF 2D IMAGE IN MATLAB

J. Kukal [1], *A. Procházka* [1], *D. Majerová* [2]

[1] Institute of Chemical Technology, Prague, [2] Czech Technical University in Prague

Abstract

Human eye vision is oriented both to global scene, and to small details. We are very sensitive to context information. The context is the difference between intensities of image pixel and their neighborhood. So, the trivial way to context extraction is based on linear high-pass filtering of given 2D image. The negative role of any pixel intensity noise is in extreme context information which is send into human eyes. The main idea of this paper is based on converting the original noised image into context domain, then nonlinear robust 2D filtering of context information and finally the inversion of context information. Various robust filters with various values of parameter $w \in (0, 1)$ are compared using ΔSNR and ΔSNR*. All the calculations were performed in the MATLAB environment.

1 Image Context

A context in image plays cardinal role in human vision. Local context means the difference between neighbor pixels in the case of digital image processing. There are many approaches and digital filters whose enable to extract local context information. LoG and DoG filters are only two examples from the class of linear high-pass filters. Supposing that the resulting filter belongs to the same class, we would like to construct contextual filter with the smallest possible neighborhood.

The simple way of pure context making can be derived from the pixel and its side neighborhoods $u_{i,j} = \frac{1}{2}x_{i,j} - \frac{1}{8}\left(x_{i,j+1} + x_{i,j-1} + x_{i+1,j} + x_{i-1,j}\right)$. Using 2D Fourier transform (FFT) we obtain

$$\mathrm{U}(\omega_1, \omega_2) = \mathrm{F}(\omega_1, \omega_2)\mathrm{X}(\omega_1, \omega_2)$$

with transform function $\mathrm{F}(\omega_1, \omega_2) = \frac{1}{2} - \frac{1}{4}(\cos\omega_1 + \cos\omega_2) \in [0, 1]$. However, the context making function C is not invertible because of $\mathrm{F}(0, 0) = 0$. But we can use a kind of pseudoinversion:

$$\mathrm{F}^{+}(\omega_1, \omega_2) = \begin{cases} 1/\mathrm{F}(\omega_1, \omega_2) & \text{for } \omega_1^2 + \omega_2^2 > 0, \\ 0 & \text{for } \omega_1 = \omega_2 = 0. \end{cases}$$

This approach can be called *pure context making* and it has two disadvantages: high sensitivity to noise and shifting of average image intensity in the case of context inversion. So, it is necessary to develop a compromise between the pure context and original image with parameter $w \in (0, 1)$ and invertible transfer function $\mathrm{F}(\omega_1, \omega_2) = 1 - \frac{w}{2}\left(1 + \frac{1}{2}(\cos\omega_1 + \cos\omega_2)\right)$. Now we have $1 - w \le \mathrm{F}(\omega_1, \omega_2) \le 1$ and the problems with transfer inversion are eliminated. The sensitivity to context inversion is then $(1 - w)^{-1}$. There are two special cases:

- $w \to 0^+$ for original instead of context,
- $w \to 1^-$ for pure context.

The best results of context denoising are obtained for the range $w \in [0.3, 0.5]$. Thus, the value $w = 0.4$ is a kind of compromise one. The properties of compromise image are demonstrated in the Figs. 1-3, where the original image is compared with its pure and compromise contexts.

2 Context denoising

The operations of the context making and context inversion will help us to develop a new kind of nonlinear filter. The new filter will make context denoising instead of the original image denoising. Let $\mathbf{X}, \mathbf{U}, \mathbf{V}, \mathbf{Y} \in \mathbb{R}^{m \times n}$ be matrices of original 2D image, its 2D context, denoised 2D context, and denoised 2D image where $m, n \in \mathbb{N}$ are image sizes. Let $\mathrm{C}, \mathrm{R} : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$ be context making and robust filtering functions where C is invertible. The general scheme of robust filtering in complex domain can be described as three step algorithm:

$$\mathbf{U} = \mathrm{C}(\mathbf{X}),\ \mathbf{V} = \mathrm{R}(\mathbf{U}),\ \mathbf{Y} = \mathrm{C}^{-1}(\mathbf{V})$$

or as composed denoising formula

$$\mathbf{Y} = \mathrm{C}^{-1}(\mathrm{R}(\mathrm{C}(\mathbf{X}))).$$

The meaning of the image denoising is depicted in the Figs. 1, 4 for the special case of WBES nonlinear robust filter. The meaning of the context denoising is depicted in the Figs. 3, 5 for WBES filter with $w = 0.4$. The denoised context image was then converted via context inversion to resulting image with denoised context as shown in the Fig. 6. If we compare the Figs. 4 and 6, we can recognize the softness of robust filtering in context domain. There is an analogy between traditional SOS (Save Our Souls) and this new SOE (Save Our Eyes) approaches.

3 Nonlinear filtering

The nonlinear robust filtering can be based on L, M, R-statistical estimates. L-estimates are based on statistical sample sorting and linear combination of sorted values. Median, quasimedian, pivot halfsum, BES and trimmed average are traditional examples. Complete average is also L-estimate but it is linear and non-robust. M-estimates are based on maximization of statistical likelihood for various models. Median, Tukey biweight, Huber and average are examples of M-estimates. Very sophisticated R-estimates are derived from rank based statistical tests. Hodges-Lehmann median with Walsh list inside is a good example of R-estimate.

Let $n \in \mathbb{N}$ and $(x_1, x_2, \ldots, x_n)$ be a list. Then the Walsh list is defined as list which contains each element $\left(\frac{x_i + x_j}{2}\right)$ such that $i \leq j$.

The median of Walsh list is called Hodges-Lehmann median.

Let $\vec{x} = (x_1, x_2, \ldots, x_n)$ be a list ($n \in \mathbb{N}$). Then the best easy systematic estimation (BES) is defined as

$$\mathrm{BES}(\vec{x}) = \frac{1}{4} \cdot \left(x_{(\lceil \frac{n}{4} \rceil)} + x_{(\lfloor \frac{n+1}{2} \rfloor)} + x_{(\lceil \frac{n+1}{2} \rceil)} + x_{(\lfloor \frac{3n+4}{4} \rfloor)} \right).$$

WBES is defined as BES estimation of Walsh list.

There are also relationships among filtering approaches. The robustness increases in five series:

- average, trimmed average, median;
- pivot halfsum, quasimedian, median;
- average, Huber, median;
- average, WBES, BES, median;
- average, Hodges-Lehmann median, median.

All the discussed methods except averaging can be used for robust contextual denoising. Standard box mask 3×3 was used for the filter realization. Source Matlab codes are introduced in the Figs. 8-10.

4 Denoising quality

Let $\mathbf{X}, \mathbf{U}, \mathbf{V}, \mathbf{Y} \in \mathbb{R}^{m \times n}$ be matrices of original 2D image, its 2D context, denoised 2D context, and denoised 2D image. Let $\mathbf{I}, \mathbf{C} \in \mathbb{R}^{m \times n}$ be an ideal image and its context. Then the quality of contextual denoising can be studied in traditional sense as $\Delta\text{SNR} = \text{SNR}_4 - \text{SNR}_1$ or in context domain as $\Delta\text{SNR}^* = \text{SNR}_3 - \text{SNR}_2$ where

$$\begin{aligned}
\text{SNR}_1 &= 10 \log_{10} \frac{\text{var}(\mathbf{I})}{\text{var}(\mathbf{X} - \mathbf{I})} \\
\text{SNR}_2 &= 10 \log_{10} \frac{\text{var}(\mathbf{C})}{\text{var}(\mathbf{U} - \mathbf{C})} \\
\text{SNR}_3 &= 10 \log_{10} \frac{\text{var}(\mathbf{C})}{\text{var}(\mathbf{V} - \mathbf{C})} \\
\text{SNR}_4 &= 10 \log_{10} \frac{\text{var}(\mathbf{I})}{\text{var}(\mathbf{Y} - \mathbf{I})}
\end{aligned}$$

A set of nine nonlinear filters plus linear one for reference was used with 3×3 mask and $w = 0.4$. The original image of coins was corrupted by gaussian noise, which is not too optimistic for nonlinear filtering. The result of testing are collected in the Tab. 1. The filtering approaches were tested without context ($w \to 0^+$) and in context domain ($w = 0.4$). The general observation is that traditional ΔSNR criterion decreases with context parameter $w \in (0, 1)$ but the context like ΔSNR^* increases with context parameter $w \in \left(0, \frac{1}{2}\right\rangle$. So, the contextual filtering is not better than traditional in ΔSNR but it is better in ΔSNR^* (better for our eyes).

5 Conclusion

A new kind of nonlinear robust contextual filters was developed. Various filters were computed with 3×3 mask for $w = 0.4$ using ΔSNR and ΔSNR^*. The best one of them is WBES (Best Easy Systematic Estimation on Walsh list) which is also acceptable in the case of gaussian noise. All the calculations were made in the Matlab environment. Source codes are included.

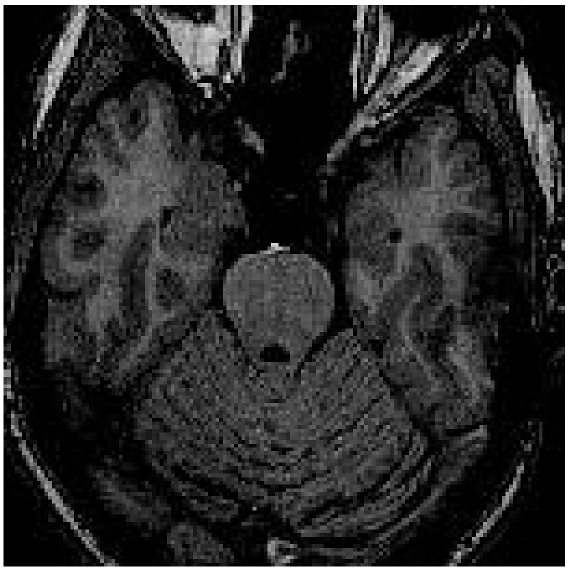
Figure 1: Original image ($w \to 0^+$)

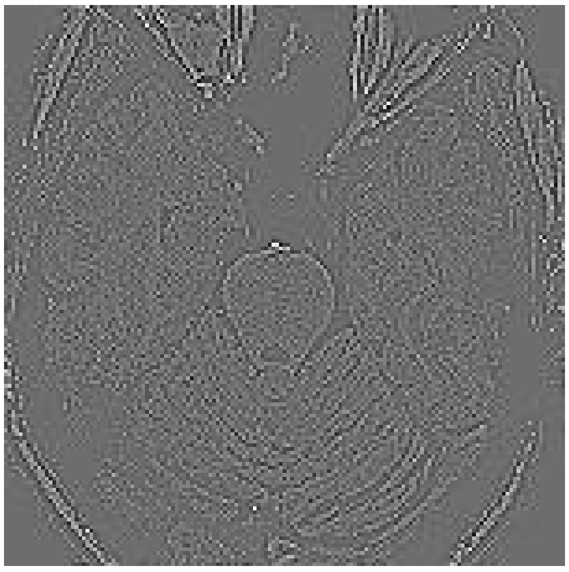
Figure 2: Pure context ($w \to 1^-$)

Table 1: DENOISING QUALITY

Type	*Filter* 3×3	$w \to 0^+$	$w = 0.4$	
		ΔSNR	ΔSNR	ΔSNR*
L,M	average	9.439	9.420	10.978
L,R	WBES	9.210	9.165	10.678
L	trimmed	9.118	9.014	10.534
R	Hodges-Lehmann	8.945	8.853	10.317
M	Huber	8.664	8.411	9.902
L	BES	8.538	8.290	9.744
L	pivot halfsum	8.425	8.264	9.610
L	quasimedian	8.115	7.756	9.177
M	Tukey biweight	7.690	7.268	8.667
L,M	median	7.100	6.609	7.923

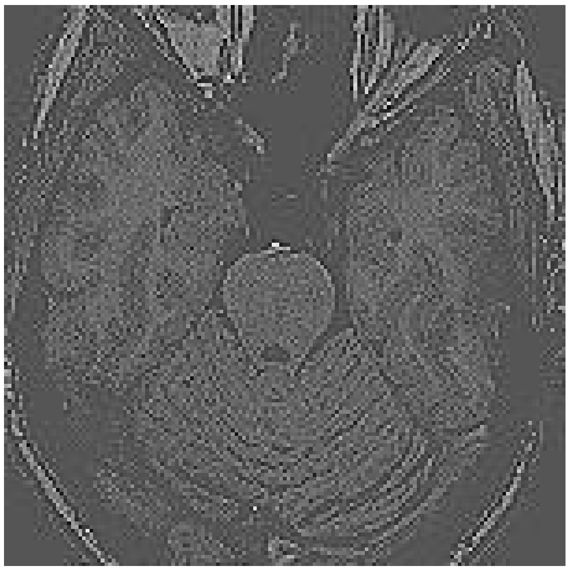
Figure 3: Compromise context ($w = 0.4$)

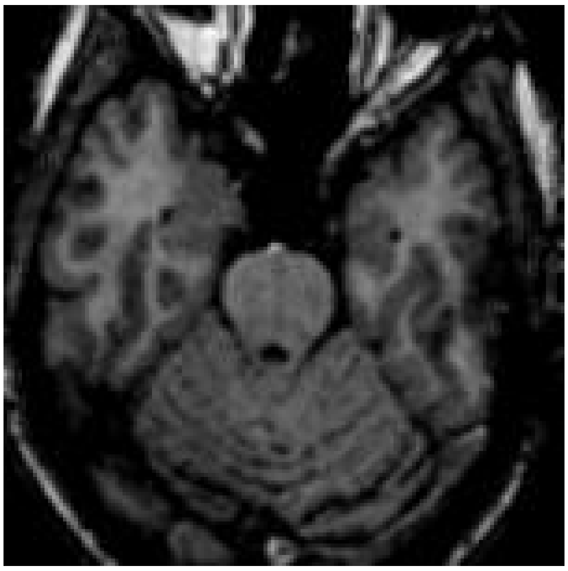
Figure 4: Denoised image (WBES, $w \to 0^+$)

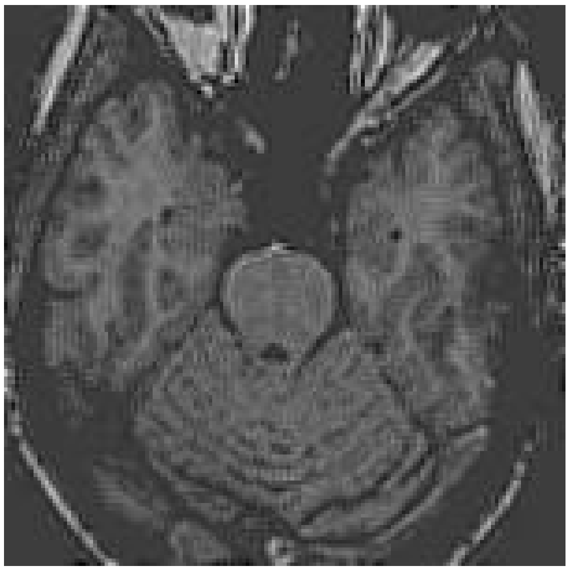
Figure 5: Denoised context (WBES, $w = 0.4$)

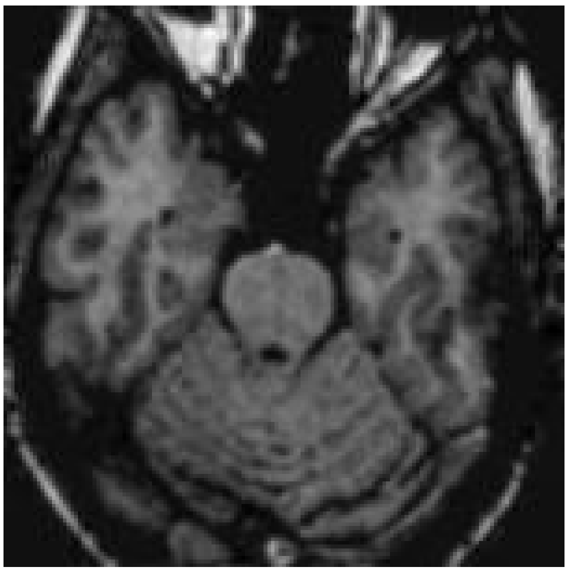
Figure 6: Image with denoised context

Acknowledgments

The work has been supported by the fund No. MSM 6046137306 of Ministry of Education of the Czech Republic. This support is very gratefully acknowledged.

References

[1] R. Klette and P. Zamperoni. *Handbook of Image Processing Operators.* John Wiley and Sons, Chichester, 1996.

[2] J.L. Hodges and E.L. Lehmann. On medians and quasi medians. *Journal of the American Statistical Association*, 62:926–931, 1967.

[3] J.W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, New York, 1977.

[4] D. Majerová and J. Kukal. Multicriteria Approach to 2D Image De-Noising by Means of Łukasiewicz Algebra with Square Root. *Neural Network World*, 2002, 12, 4, p. 333–348.

Jaromír Kukal
Institute of Chemical Technology, Prague
Department of Computing and Control Engineering
Jaromir.Kukal@vscht.cz

Aleš Procházka
Institute of Chemical Technology, Prague
Department of Computing and Control Engineering
A.Prochazka@ieee.org

Dana Majerová
Czech Technical University in Prague
Department of Software Engineering in Economy
Dana.Majerova@dc.fjfi.cvut.cz

```
function y=IMG2DFILTER(filtername,x,mask)
% General filter for 2D image
% y=IMG2DFILTER(filtername,x,mask);
% y ......  output image matrix (m,n,h)
% filtername ...  name of basic filter:  result=filter(vector)
% x ......  input image matrix (m,n)
% mask ...  mask matrix (2*r+1,2*r+1,h)
% m ......  image height
% n ......  image width
% h ......  mask number
% r ......  mask radius
% Examples:
% x(m,n,h) without mask will produce y(1,1)
% x(1,n) with mask(1,n) will produce y(1,1)
% x(m,n,h) with mask(1,h) will produce y(m,n)
% x(m,n) with mask(2*r+1,2*r+1) will produce y(m,n)
% x(m,n) with mask(2*r+1,2*r+1,h) will produce y(m,n,h)

[m,n,h]=size(x);

if nargin==2 % absence of mask
   y=feval(filtername,x(:)); % processing of given function
   return
end

[maskm,maskn,maskh]=size(mask);

if maskm==1 & maskn>=1 & maskh==1 & m==1 & n==maskn % row mask and row image
   listsize=sum(mask);
   wlist=zeros(1,listsize);
   index=1;
   for i=1:maskn
      for j=1:mask(i)
         wlist(index)=x(i);
         index=index+1;
      end
   end
   y=IMG2DFILTER(filtername,wlist);
   return
end

y=zeros(m,n);
```

Figure 7: Nonlinear 2D filter

```
function [y,c,fc]=CONTEXT2D(x,w)
%CONTEXT2D 2D context filtering 3*3 with WBES filter
%[y,c,fc]=CONTEXT2D(x,w);
%y ....  result of filtering
%c ....  context of source
%fc ...  filtered context = context of result
%x ....  original image as double matrix (m,n)
%w ....  weigth of context [0,1] 0.4 recommended
[n,m]=size(x);
mask=ones(3,3);
ome1=(0:m-1)*2*pi/m;
ome2=(0:n-1)*2*pi/n;
[omega1,omega2]=meshgrid(ome1,ome2);
F=abs(1-w/2*(1+(cos(omega1)+cos(omega2))/2));
c=real(ifft2(fft2(x).*F));
fc=IMG2DFILTER('WBES',c,mask);
F(F==0)=inf;
y=real(ifft2(fft2(fc)./F));
```

Figure 8: Contextual 2D filter

```
function [y]=WBES(x)
xxx=x(:)';
nn=size(xxx,2);
hoho=zeros(1,nn*(nn+1)/2);
k=1;
for i=1:nn
   for j=i:nn
      hoho(k)=(xxx(i)+xxx(j))/2;
      k=k+1;
   end
end
y=BES(hoho);
```

Figure 9: WBES statistics

```
function [y]=BES(x)
x=sort(x);
n=length(x);
y=(x(ceil(n/4))+x(floor((n+1)/2))+x(ceil((n+1)/2))+x(floor((3*n+4)/4)))/4;
```

Figure 10: BES statistics