تجزیه وتحلیل داده ها در نساجی

پردازش تصویر

دکتر پدرام پیوندی

بخش پنجم

www.pedram-payvandy.com
peivandi@yazduni.ac.ir

فیلترهای غیر خطی

ordfilt2

2-D order-statistic filtering

Syntax

```
B = ordfilt2(A, order, domain)
B = ordfilt2(A, order, domain, S)
B = ordfilt2(..., padopt)
```

Description

`B = ordfilt2(A, order, domain)` replaces each element in `A` by the `order`th element in the sorted set of neighbors specified by the nonzero elements in `domain`.

`B = ordfilt2(A, order, domain, S)` where `S` is the same size as `domain`, uses the values of `S` corresponding to the nonzero values of `domain` as additive offsets.

`B = ordfilt2(..., padopt)` controls how the matrix boundaries are padded. Set `padopt` to `'zeros'` (the default) or `'symmetric'`. If `padopt` is `'zeros'`, `A` is padded with 0's at the boundaries. If `padopt` is `'symmetric'`, `A` is symmetrically extended at the boundaries.

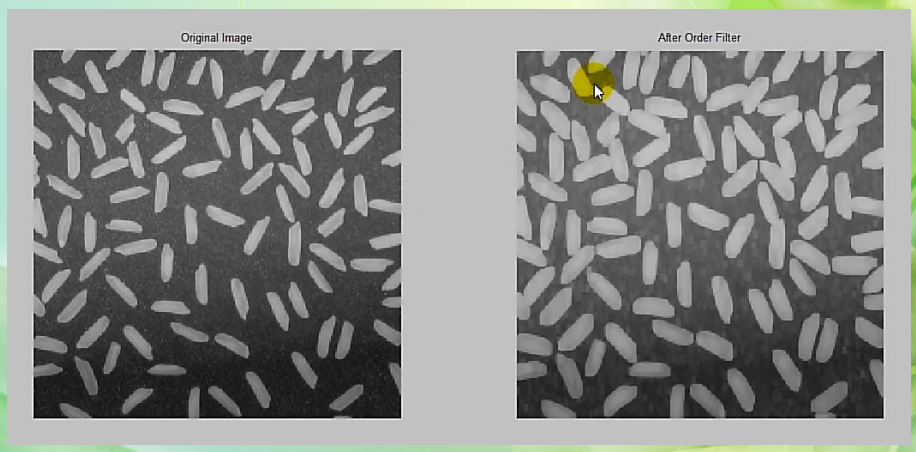
www.pedram-payvandy.com
peivandi@yazduni.ac.ir

فیلترهای غیر خطی

```
clc;
clear;
close all;

img0=imread('rice.png');

img0=im2double(img0);

M=5;
N=3;

Domain=ones(M,N);

img1=ordfilt2(img0,M*N,Domain);
```

www.pedram-payvandy.com
peivandi@yazduni.ac.ir

فیلترهای غیر خطی



www.pedram-payvandy.com
peivandi@yazduni.ac.ir

فیلترهای غیر خطی

medfilt2

2-D median filtering

Note The syntax `medfilt2(A,[M N],(Mb Nb],...)` has been removed.

Syntax

```
B = medfilt2(A, [m n])
B = medfilt2(A)
B = medfilt2(A, 'indexed', ...)
B = medfilt2(..., padopt)
```

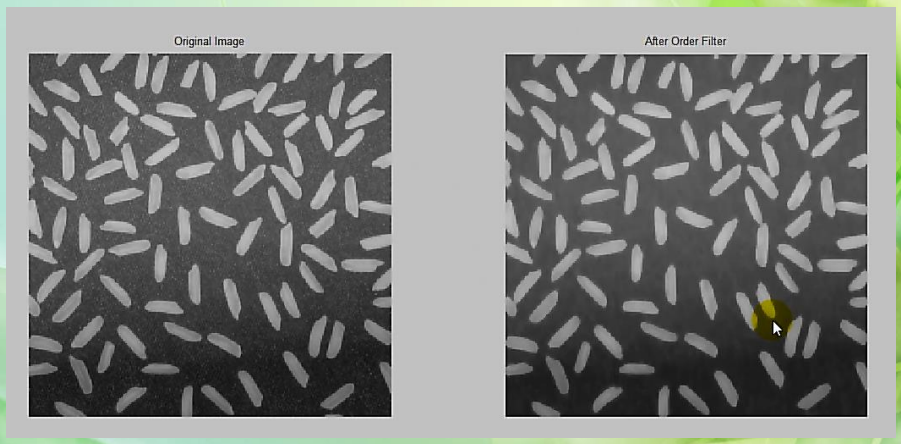
Description

Median filtering is a nonlinear operation often used in image processing to reduce "salt and pepper" noise. A median filter is more effective than convolution when the goal is to simultaneously reduce noise and preserve edges.

www.pedram-payvandy.com
peivandi@yazduni.ac.ir

5

فیلترهای غیر خطی



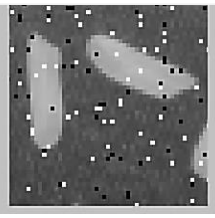
www.pedram-payvandy.com
peivandi@yazduni.ac.ir

6

فیلترهای غیر خطی

`J = imnoise(I,type)` adds noise of a given type to the intensity image `I`. *type* is a string that can have one of these values.

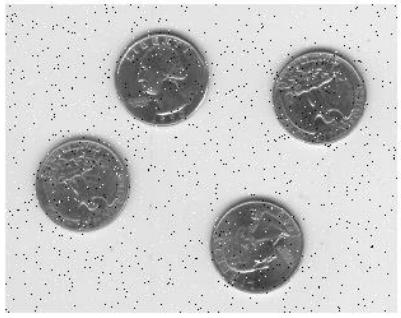
Value	Description
`'gaussian'`	Gaussian white noise with constant mean and variance
`'localvar'`	Zero-mean Gaussian white noise with an intensity-dependent variance
`'poisson'`	Poisson noise
`'salt & pepper'`	On and off pixels
`'speckle'`	Multiplicative noise



فیلترهای غیر خطی

Examples

```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
figure, imshow(I)
figure, imshow(J)
```



فیلترهای غیر خطی

$$g(x,y) = \phi\left(N(x,y)\right)$$

colfilt Column Filter

colfilt

Columnwise neighborhood operations

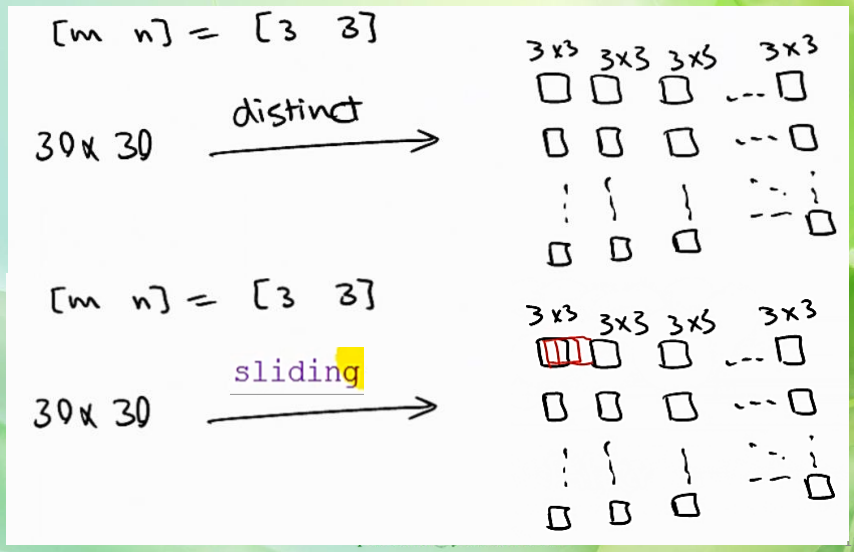
Syntax

```
B = colfilt(A,[m n],block_type,fun)
B = colfilt(A,[m n],[mblock nblock],block_type,fun)
B = colfilt(A,'indexed',...)
```


فیلترهای غیر خطی

Value	Description
`'distinct'`	Rearranges each `m`-by-`n` distinct block of `A` into a column in a temporary matrix, and then applies the function `fun` to this matrix. `fun` must return a matrix the same size as the temporary matrix. `colfilt` then rearranges the columns of the matrix returned by `fun` into `m`-by-`n` distinct blocks.
`'sliding'`	Rearranges each `m`-by-`n` sliding neighborhood of `A` into a column in a temporary matrix, and then applies the function `fun` to this matrix. `fun` must return a row vector containing a single value for each column in the temporary matrix. (Column compression functions such as `sum` return the appropriate type of output.) `colfilt` then rearranges the vector returned by `fun` into a matrix the same size as `A`.

فیلترهای غیر خطی



فیلترهای غیر خطی

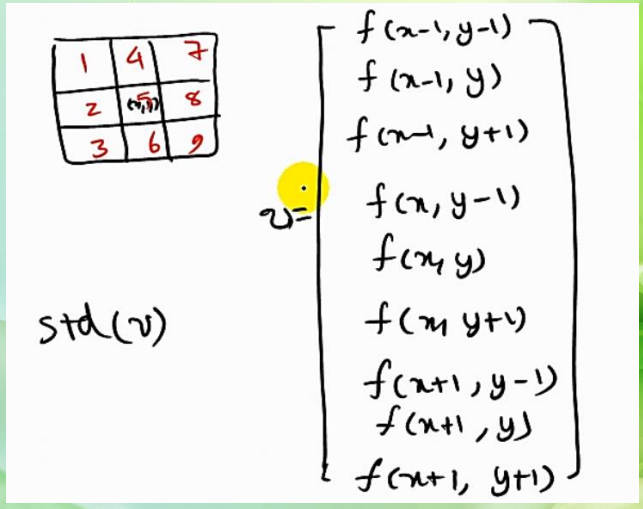
$$x_1, x_2, \cdots, x_N$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \longrightarrow \text{std}$$

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \longrightarrow \text{mean}$$

www.pedram-payvandy.com
peivandi@yazduni.ac.ir

فیلترهای غیر خطی



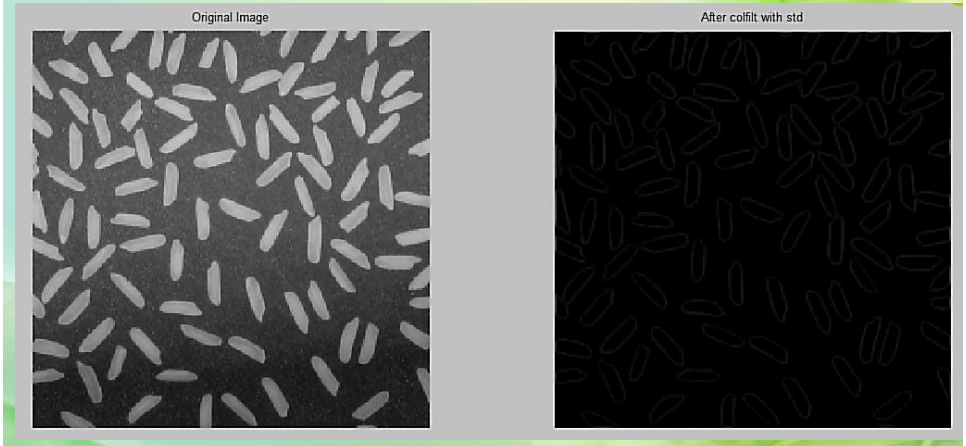
www.pedram-payvandy.com
peivandi@yazduni.ac.ir

فی

```
clc;
clear;
close all;

img0=imread('rice.png');

img0=im2double(img0);

M=3;
N=3;

img1=colfilt(img0,[M N],'sliding',@std);
```

```
img0=imread('rice.png');

img0=im2double(img0);

M=3;
N=3;

img1=colfilt(img0,[M N],'distinct',@sort);
```

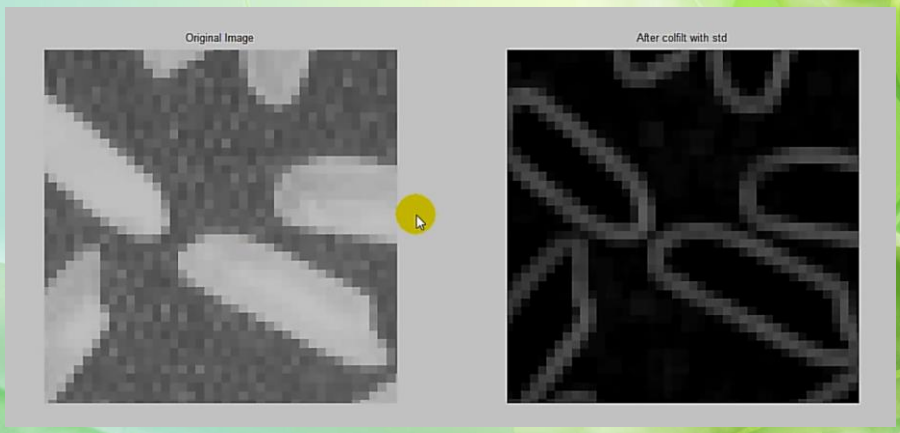
فیلترهای غیر خطی



www.pedram-payvandy.com
peivandi@yazduni.ac.ir

15

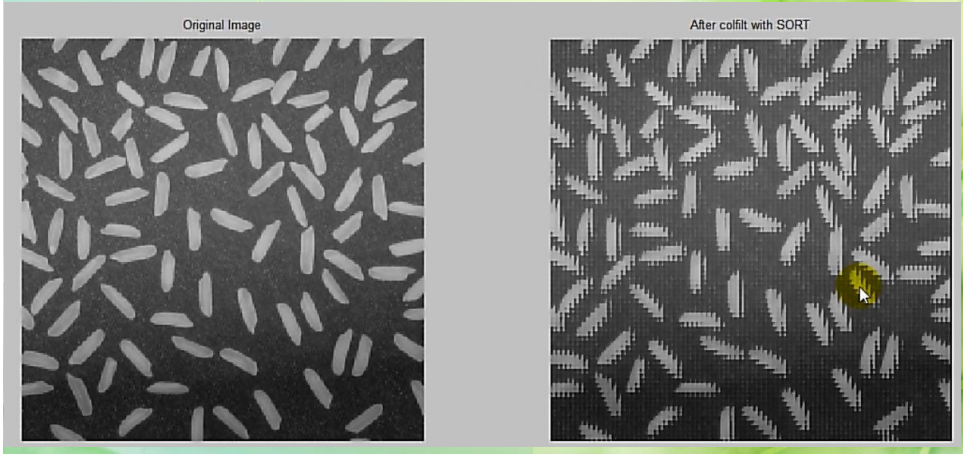
فیلترهای غیر خطی



www.pedram-payvandy.com
peivandi@yazduni.ac.ir

16

فیلترهای غیر خطی



www.pedram-payvandy.com
peivandi@yazduni.ac.ir

17

فیلترهای غیر خطی

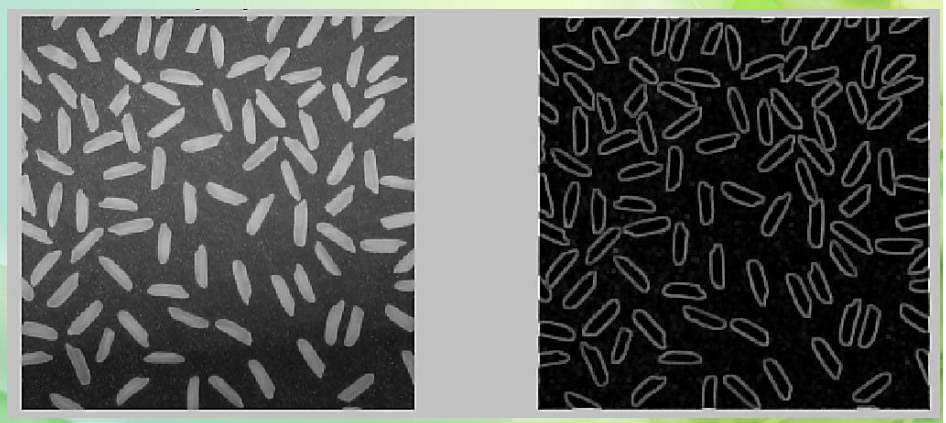
www.pedram-payvandy.com
peivandi@yazduni.ac.ir

18

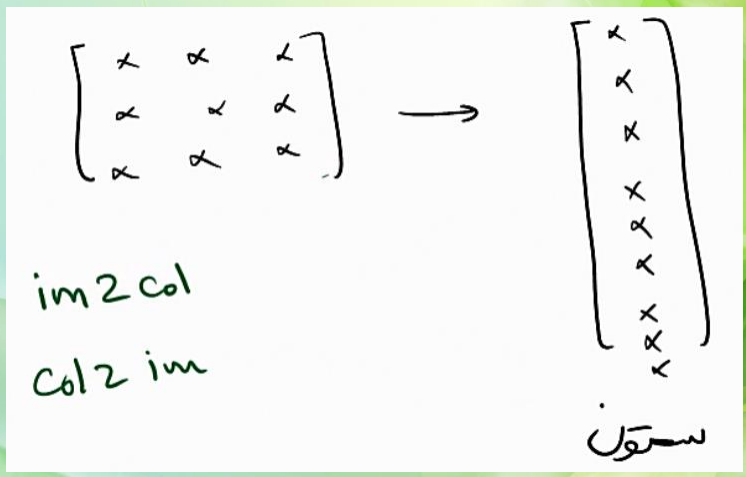
فیلترهای غیر خطی

```
clc;
clear;
close all;

img0=imread('rice.png');

img0=im2double(img0);

M=3;
N=3;

MyFun=@(X) max(X)-min(X);

img1=colfilt(img0,[M N],'sliding',@MyNonlinearFilter);
```


فیلترهای غیر خطی



فیلترهای غیر خطی



فیلترهای غیر خطی

im2col

Rearrange image blocks into columns

Syntax

```
B = im2col(A,[m n],block_type)
B = im2col(A,'indexed',...)
```

Description

`B = im2col(A,[m n],block_type)` rearranges image blocks into columns. `block_type` is a string that can have one of these values. The default value is enclosed in braces ({ }).

Value	Description
`'distinct'`	Rearranges each ***distinct*** `m`-by-`n` block in the image `A` into a column of `B`. `im2col` pads `A` with 0's, if necessary, so its size is an integer multiple of `m`-by-`n`. If `A = [A11 A12; A21 A22]`, where each A_{ij} is `m`-by-`n`, then `B = [A11(:) A12(:) A21(:) A22(:)]`.
`{'sliding'}`	Converts each ***sliding*** `m`-by-`n` block of `A` into a column of `B`, with no zero padding. `B` has `m*n` rows and contains as many columns as there are `m`-by-`n` neighborhoods of `A`. If the size of `A` is `[mm nn]`, then the size of `B` is `(m*n)`-by-`((mm-m+1)*(nn-n+1))`.

فیلترهای غیر خطی

```
>> size(img0)

ans =

   256   256

>> C=im2col(img0,[3 3],'distinct');
>> size(C)

ans =

           9        7396

>> numel(C)

ans =

       66564
>> numel(img0)

ans =

       65536
```

```
>> C=im2col(img0,[3 3],'sliding');
>> size(C)

ans =

           9       64516
```

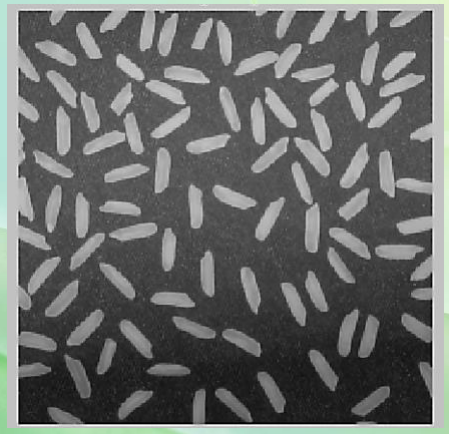
```
>> 65536-255*4

ans =

       64516
```

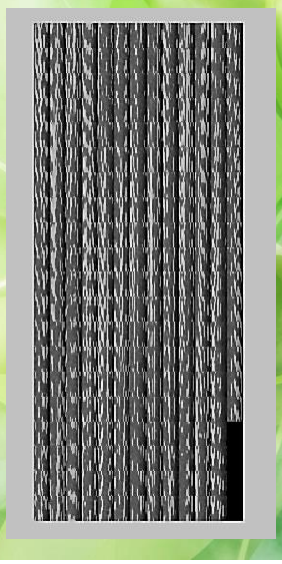
www.pedram-payvandy.com
peivandi@yazduni.ac.ir

23

فیلترهای غیر خطی

```
>> C=im2col(img0,[20 20],'distinct');
```





www.pedram-payvandy.com
peivandi@yazduni.ac.ir

فیلترهای غیر خطی

nlfilter

General sliding-neighborhood operations

Syntax

```
B = nlfilter(A, [m n], fun)
B = nlfilter(A, 'indexed',...)
```

bestblk

Determine optimal block size for block processing

Syntax

```
siz = bestblk([m n],k)
[mb,nb] = bestblk([m n],k)
```

Description

`siz = bestblk([m n],k)` returns, for an `m`-by-`n` image, the optimal block size for block processing. `k` is a scalar specifying the maximum row and column dimensions for the block; if the argument is omitted, it defaults to 100. The return value `siz` is a 1-by-2 vector containing the row and column dimensions for the block.

`[mb,nb] = bestblk([m n],k)` returns the row and column dimensions for the block in `mb` and `nb`, respectively.

25

فیلترهای غیر خطی

blockproc

Distinct block processing for image

Syntax

```
B = blockproc(A,[M N],fun)
B = blockproc(src_filename,[M N],fun)
B = blockproc(adapter,[M N],fun)
blockproc(...,Name,Value,...)
```

Description

`B = blockproc(A,[M N],fun)` processes the image `A` by applying the function `fun` to each distinct `M`-by-`N` block of `A` and concatenating the results into `B`, the output matrix. `fun` is a function handle to a function that accepts a ***block struct*** as input and returns a matrix, vector, or scalar `Y`. For example, `Y = fun(block_struct)`. (For more information about a ***block struct***, see the Definition section below.) For each block of data in the input image, `A`, `blockproc` passes the block in a ***block struct*** to the user function, `fun`, to produce `Y`, the corresponding block in the output image. If `Y` is empty, `blockproc` does not generate any output and returns empty after processing all blocks. Choosing an appropriate block size can significantly improve performance. For more information, see Choosing Block Size in the Image Processing Toolbox™ documentation.