Filters

نسیم فانی

اطلاعات گزارش

تاریخ: 99/8/29

واژگان کلیدی:
Median Filter
Box Filter
Gaussian Filter
salt-and-pepper
noise

چکیده

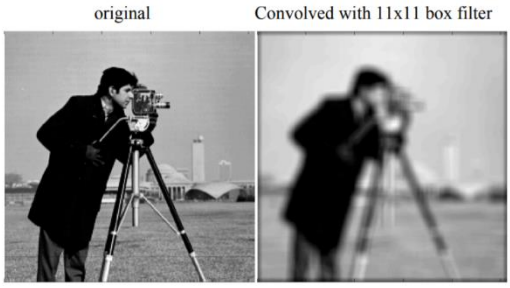
در این تمرین به بررسی فیلترهای مختلف (Box Filter, Median Filter,..)، مزایا و معایب استفاده از آن‌ها و تاثیرشان بر روی تصاویر مختلف می‌پردازیم.

1-مقدمه

نوشتار حاضر، به بررسی روش‌های اعمال و پیاده‌سازی فیلتر-های مختلف، بهبود کیفیت تصاویر و حذف نویز از تصاویر می‌پردازد.

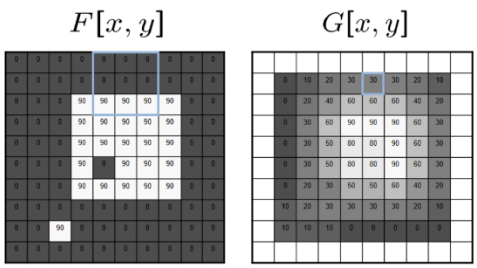
2-توضیحات تکنیکال

3.1.1 فیلتر جعبه در اصل نوعی فیلتر تصویری با پیکسل‌های اطراف است. با اعمال این فیلتر به نوعی میانگین بدون وزن و بدون در نظر گرفتن فاصله هر همسایه بین پیکسل‌های اطراف بر روی پیکسل مورد نظر اعمال می‌شود و این میانگین تصویر را blur می‌کند، جزئیات ریز تصویر را کاهش می دهد و لبه-ها را تا حد زیادی تضعیف می‌کند. در تصویر زیر این موضوع به راحتی قابل رویت ا ست.



با عبور یک فیلتر جعبه که کرنلی با مجموع مقادیر ۱ است و مقدار هر خانه را با میانگین مقدار خانه‌های همسایه‌اش جایگزین می‌کند، می‌توان تصویر را محو کرد.

این کرنل تغییرات ناگهانی در رنگ تصویر را ملایم‌تر می‌کند:



همچنین با افزایش سایز کرنل می‌توان میزان تاری تصویر را افزایش داد:

به طور کلی می‌توان گفت حساسیت این فیلتر به نویز می‌تواند تغییر زیادی را در میانگین به وجود بیاورد که خود موجب خراب شدن تصویر می‌شود.

هموارسازی با این فیلترها را نمی‌توان ابدا با لنزهای defocused مقایسه کرد.

بارزترین تفاوت در این است که یک نقطه واحد از نور که در لنزهای فوکوس شده مشاهده می شود ، مانند یک لکه فازی به نظر می رسد. ولی فرآیند میانگین گیری مقدار کمی تصویر را مربعی می کند.[1]

یکی دیگر از دلایلی که این فیلتر، فیلتر خوبی نیست، زیاد بودن حجم محاسبات(ضرب) است که با افزایش سایز پنجره به صورت نمایی زیاد می‌شود.

در یک فیلتر جعبه با اندازه پنجره 200 برا یمحاسبه‌ی هر پیکسل نیاز به انجام 40000 ضرب داریم و در یک تصویر با اندازه‌ی M*N تعداد کل ضرب‌ها 40000*M*N خواهد شد که برای یک تصویر کوچک هم محاسبات زیادی خواهد بود .

3.1.2 همان‌طور که در قسمت قبل گفته شد، این فیلتر از طریق اعمال یک میانگین بدون وزن برروی همسایه‌های هر پیکسل عمل می‌کند. با اعمال مجدد آن مقادیر پیکسل‌ها کم کم به هم نزدیک‌تر شده و موجب تارتر شدن بیشتر تصویر و ضعیف شدن شدیدتر لبه‌ها می‌شود. این عمل را اگر بیشتر تکرار کنیم، تمام پیکسل‌ها مقداری مساوی خواهند گرفت و دیگر اعمال میانگین بر آن‌ها تاثیری نخواهد داشت. اگر تصویر ما کلیت تیره داشته باشد تصویر نهایی بعد از تکرار زیاد این فیلتر یک تصویر یکنواخت و تیره(خاکستری رنگ) خواهد بود.

در مورد نویزها نیز اعمال زیاد این فیلتر نتیجه‌ای منفی دارد.

3.1.3 برای پیاده سازی فیلتر جعبه ابتدا لازم است تصویر را پدینگ صفر بدهیم.

سپس از طریق عمل ضرب فیلتر را با تصویر کانوالو می‌کنیم.

در نهایت به اندازه‌ی سایز تصویر اولیه‌مان از مرکز نتیجه‌ی حاصل جدا می‌کنیم.

همانطور که گفته شد با تکرار این فیلتر، تصویر تارتر می‌شود اما از یه حدی به بعد، تصویر دیگر تار تر نمی‌شود.

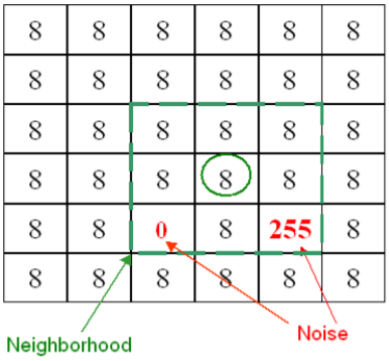
3.2.1 برای اضافه کردن نویز salt-and-pepper از تابع imnoise به صورت زیر استفاده می‌کنیم :

```
imnoise(,'salt & pepper',density);
```

این تابع به صورت رندوم تعدادی از پیکسل‌های تصویر را سفید و تعدادی را سیاه می‌کند. تراکم این نقاط سیاه و سفید را می‌توان با تغییر مقدار density تغییر داد.

واضح است که با افزایش density، تراکم نویز بیشتر می‌شود.

حال می‌خواهیم با کمک median filter به بهبود کیفیت تصاویر و حذف نویز آن‌ها بپردازیم. فیلتر میانه یک فیلتر آماری غیرخطی است. این فیلتر عناصر داخل پنجره را مرتب می‌کند و مقدار میانی آن را به عنوان مقدار پیکسل مورد نظر قرار می‌دهد در این صورت مقادیر خیلی زیاد یا خیلی کم (که همان نویز فلفل و نمک هستند) از تصویر ما حذف می‌شود. به عنوان مثال با در نظر گرفتن یک پنجره‌ی 3*3 و اعمال فیلتر میانه داریم:



مقادیر پیکسل را به ترتیب افزایش طبقه بندی می‌کنیم:

0 ، 8 ، 8 ، 8 ، **8** ، 8 ، 8 ، 255

مقدار متوسط 8 است. و همان‌طور که مشاهده می‌شود مقادیر درخشندگی شدید 0 و 255 تاثیری بر مقدار خروجی توسط ندارد. پس برای نویز فلفل نمکی، فیلتر کارآمدی است.

اگر تراکم نویز ما زیاد باشد و سایز پنجره‌ی فیلتر را کوچک در نظر بگیریم ممکن است برخی از نویزها در تصویرمان حذف نشود. دلیل این امر این است که چون تراکم نویز در تصویرمان زیاد است در یک پنجره‌ی کوچک ممکن است میانه یک نویز باشد. برای مثال:

0 ، 0 ، 0 ، 0 ، **0** ، 8 ، 8 ، 255 ، 255

در این‌صورت نویز به عنوان میانه انتخاب می‌شود.

بالعکس در تراکم نویز پایین بهتر است از فیلتر میانه با اندازه‌ی پنجره‌ی کوچک استفاده کنیم تا تصویر خروجی‌مان دقیق‌تر و با جزئیات بیشتر باشد و با انجام محاسبات کمتر سرعت بیشتری داشته باشیم ضمن اینکه کیفیت کار را کاهش نداده ایم.

برای بررسی و مقایسه بهتر تصاویر حاصل، مقدار mse را برای هر تصویر محاسبه می‌کنیم.

3.2.2 ابتدا نویز گوسین را با به صورت زیر به تصویرمان اضافه می‌کنیم:

imnoise(I,'gaussian',0, variance)

Gaussian Noise نوعی نویز آماری است که دارای عملکرد تراکم احتمال برابر با توزیع نرمال است که به آن توزیع Gaussian نیز می‌گویند. برای تولید این نویز تابع Gaussian به صورت Random به تصویر اضافه می شود.

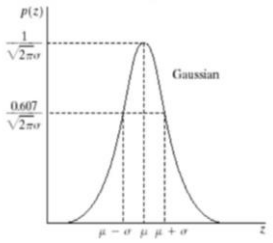
تابع گوسین به صورت زیر بدست می‌آید:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-u)^2/2\sigma^2}$$

z: gray level

μ: mean of random variable z

σ^2: variance of z



حال می‌خواهیم به کمک دو فیلتر میانه و Box Filter به بهبود کیفیت این تصاویر بپردازیم.

سپس به مقادیر mse را برای هر تصویر محاسبه کرده و از روی مقادیر آن دو فیلتر را با یکدیگر مقایسه می‌نماییم.

3.3.1 تصویر تار و نویزدار cameraman را در نظر می-گیریم.

با اعمال فیلتر unsharp می‌توان لبه‌های این تصویر را واضح‌تر نمود تا کمتر تار باشد.
همچنین برای حذف نویز از این تصویر می‌توان از فیلترهای گوناگونی مانند box filter، فیلتر گوسین و... استفاده کرد.
3.4.1 لبه یابی در واقع مجموعه عملیات ریاضی می باشد که به کمک آنها می توان نقاطی از تصویر که در آنها روشنایی بطور شدید تغییر می کند را شناسایی کرد. لبه ها معمولا بصورت خطوطی که دارای انحنا هستند مشخص می شوند.

از لبه یابی می توان برای تشخیص تغییرات شدید در روشنایی که معمولا نشانه رویدادی مهم یا تغییر در محیط است، استفاده کرد. همچنین می توان از لبه یابی در object recognition (تشخیص اشیا) و segmentation(جدا سازی عکس و تبدیل آن به چند عکس) و بینایی ماشین استفاده کرد.
همانطور که گفته شد لبه ها محلی هستند که در آن تابع شدت روشنایی دچار تغییرات شدید می شود. پس لبه ها در واقع همان قله ها در تابع مشتق اول است.
برای اینکه مشتق اول را روی شکل پیاده سازی کنیم، از یک ماتریس استفاده می کنیم و آن را روی پیکسل ها حرکت می دهیم. همچنین باید یک treshold(آستانه) تعیین کنیم که بیشتر از چه مقدار تغییر در شدت روشنایی را به عنوان لبه در نظر بگیریم.

سه فیلتر داده شده را از طریق correlate کردن فیلترها با تصاویر به آن‌ها اعمال می‌کنیم و انتظار داریم لبه‌های عمودی تصویر detect شوند.
فلیتر a یک فیلتر یک بعدی است و فقط اختلاف بین دو پیکسل کنار را در نظر می‌گیرد و به همین دلیل خطای آن زیاد است و ممکن است محلی را که لبه نیست، به عنوان لبه تشخیص دهد. همچنین به دلیل کوچکی پنجره‌ی اسن فیلتر، لبه‌های حاصل از آن نازک هستند.
فیلتر b نسبت به فیلتر اول از دقت بیشتری برخوردار است اما تفاوتی در وزن پیکسل‌های همسایه‌ی کناری و پیکسل-های همسایه قطری قائل نمی-شود.
فیلترc مانند فیلتر b دوبدی بوده و دقت بیشتری از فیلتر اول دارد. علاوه بر این وزن بیشتری را برای پیکسل‌های کناری نسبت به قطری قائل شده است.
دو فیلتر b وc لبه‌های قطورتری نسبت به فیلتر اول ایجاد می‌کنند.
برای کورلیت کردن فیلتر از عمل ضرب آرایه‌ای استفاده می‌کنیم.

3.4.2 این فیلتر یکی از نخستین روش های تشخیص لبه تصویر است که ازدو ماتریس ۲*۲ استفاده می کند. این فیلتربرای ماسک های دوبعدی با اولویت قطر است و مبتنی بر پیاده سازی تفاضل های قطری هستند. به این صورت که فیلتر اول تخمین مشتق افقی و فیلتر دوم تخمین مشتق عمودی را حساب می‌کند.
دلیل اصلی استفاده از عملگر Roberts Cross محاسبه سریع آن است. برای تعیین مقدار هر پیکسل خروجی فقط چهار پیکسل ورودی باید بررسی شود و در محاسبه فقط از تفریق و جمع استفاده می شود. علاوه بر این هیچ پارامتری برای تنظیم وجود ندارد.
معایب اصلی آن این است که از آنجا که از چنین هسته کوچکی استفاده می کند ، به نویز بسیار حساس است.

همچنین پاسخ های بسیار ضعیفی به لبه های اصلی می-دهد مگر اینکه خیلی تیز باشند.

نحوه محاسبه:

Horizontal Filter

1	0
0	-1

Verticle Filter

0	1
-1	0

برای مثال اگر یه پنجره 2*2 داشته باشیم:

p1	p2
p3	p4

به طوری که بخواهیم فیلتر را بر p1 اعمال کنیم و p2 را برابر [y][x+1]، p3 را برابر [x][y+1] و... در نظر بگیریم. دراین‌صورت داریم:

pixel = SQRT((X*X)+(Y*Y))
where X = abs(p1-p4) and Y = abs(p2-p3)

برای ساده‌تر دن محاسبات می‌توان از روش زیر استفاده کرد:

pixel = abs(p1-p4)+abs(p2-p3)

در پیاده سازی ابتدا تصویر را فیلتر کرده و سپس با اعمال یک treshhold لبه‌ها را جداسازی می-کنیم.

3.5.1 ماتریس 'I همان کرنل مربوط به فیلتر محوی (Blur) است که تصویر را بدون جزئیات تولید می‌کند؛ با تفریق این ماتریس از خود تصویر جزئیات تصویر اصلی حاصل می‌شود؛ حال با اضافه کردن این ماتریس به خود تصویر می‌توان جزئیات تصویر را به آن افزود و تصویری با جزئیات نمایان‌تر شده تولید کرد.

با افزودن ضریب آلفا میتوان میزان تاثیر این جزئیات را تنظیم نمود.

3-شکل‌ها، جدول‌ها و روابط (فرمول‌ها)

3.1.3 با چندین بار اعمال این فیلتر داریم:

تصویر1 – فیلتر جعبه‌ای

3.2.1 نتیجه اعمال این نویز با سه تراکم 0.05، 0.1 و 0.2 به صورت زیر است:

تصویر2 – نویز فلفل و نمک

تصاویر زیر نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی پنجره‌های 3، 5، 7 و 9 را بر روی هر سه تصویر نویز دار (به ترتیب) است:

تصویر3 – اعمال فیلتر میانه بر روی سه تصویر در (تصویر2)

MSE Report

	3*3	5*5	7*7	9*9
0.05	29.10	58.48	90.52	120.47
0.1	39.39	65.16	98.31	131.36
0.2	90.13	90.54	137.06	181.31

3.2.2 تصاویر زیر نتایج حاصل از اعمال نویز گوسین با واریانس‌های 0.01، 0.05 و 0.1 است:

اعمال فیلترها بر روی تصاویر:

فیلتر میانه:

تصویر3 – نویز گوسین

تصویر4 – فیلتر میانه بر روی نویز گوسین

فیلتر جعبه:

تصویر5 – فیلتر جعبه‌ای بر روی نویز گوسین

MSE Report

فیلتر میانه:

	3*3	5*5	7*7	9*9
0.01	143.78	111.56	131.01	158.50
0.05	581.63	301.56	260.67	270.53
0.1	1101.21	516.86	399.85	381.76

فیلتر جعبه:

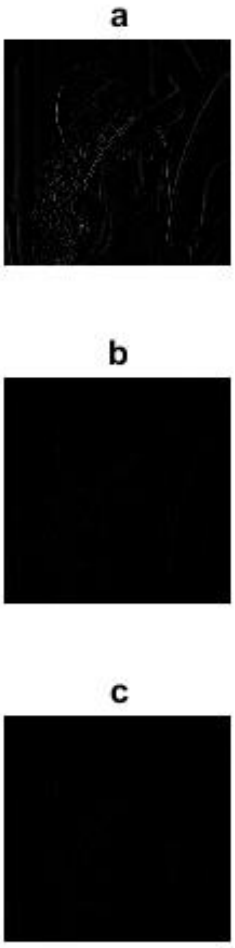
	3*3	5*5	7*7	9*9
0.01	116.81	126.81	168.60	215.94
0.05	374.74	234.48	234.47	266.06
0.1	637.19	364.78	331.60	349.56

3.3.1 نتایج حاصل از اعمال هر فیلتر بر روی تصویر cameraman که تار و دارای نویز است:

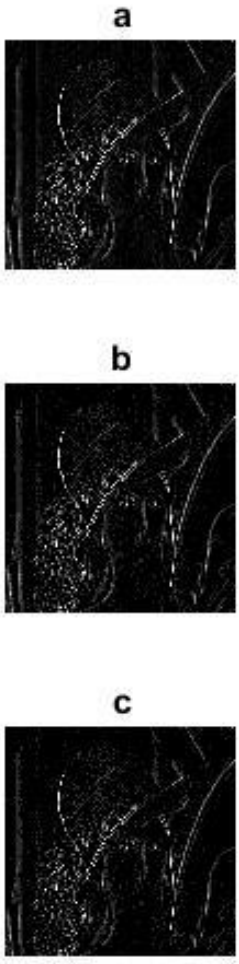
اعمال فیلتر غیرشارپ با ضریب 5 و میانگین با اندازه‌ی 3:

تصویر6

3.4.1



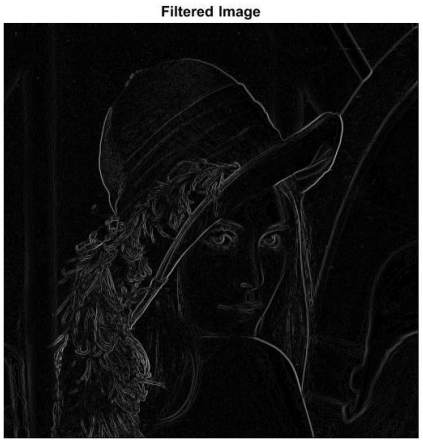
تصویر7 – لبه‌یابی

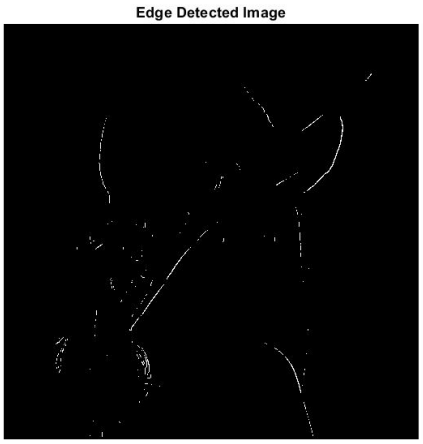


تصویر8 – بهبود کیفیت لبه‌یابی

جهت نمایش بهتر در داکیومنت با تابع imadjustکیفیت
تصاویر را بهبود می‌بخشیم:

3.4.2 نتـــایج حاصـــل از اعمـــال ایـــن فیلتـــر بـــر روی تصویر:





تصویر9 – لبه‌یاب رابرت

3.5.1 با اعمال فیلر غیرشارپ داریم با آلفاهای 0.1، 0.5، 0.9 به ترتیب داریم:

تصویر10 – فیلتر غیرشارپ

4- نتایج

به طـور کلـی فیلتـر میانـه بـرا یـاز بـین بـردن نـویز بســیار مــوثر اســت. تــاثیر آن بســتگی بــه ســایز پنجره‌ی فیلتـر و چگـالی نـویز دارد. اگـر تـراکم نـویز ما زیـاد باشـد و سـایز پنجـره‌ی فیلتـر را کوچـک در نظــر بگیــریم ممکــن اســت برخــی از نــویزهــا در

تصویرمان حذف نشود. در تراکم نویز پایین بهتر است از فیلتر میانه با اندازه‌ی پنجره‌ی کوچک استفاده کنیم تا تصویر خروجی‌مان دقیق‌تر و با جزئیات بیشتر باشد.

سایز پنجره‌ی بزرگ برای این فیلتر موجب بالاتر رفتن mse می‌شود چون تصویر را تارتر می‌کند.

فیلتر میانه روی نویز فلفل و نمک عملکرد بهتری نسبت به نویز گوسی دارد.

فیلتر میانگین عملکرد ضعیفی بر روی نویز گوسی داشته و با افزایش سایز پنجره این فیلتر، عملکرد آن ضعیف‌تر می‌شود.

پس به طور کلی می‌توان گفت فیلتر میانه برای تصاویر با نویز نمک و فلفل عملکرد بسیار بهتری نسبت به فیلتر میانه دارد .اما در تصاویر با نویز گوسین، فیلتر میانگین در بیشتر موارد بهتر است.

مقایسه تصاویر نشان می‌دهد که فیلتر روبرت لبه های جزیی بیشتری را به صورت محلی شناسایی می‌کنند. همچنین این فیلتر نسبت به نویز حساس تر است.

به لحاظ تشخیص لبه فیلترهای b وc عملکرد بهتری دارند.

در فیلتر غیرشارپ هر چه پنجره‌ی نویز را بزرگتر بگیریم، تصویر تارتر می‌شود. در نتیجه اختلاف آن با تصویر اصلی مقادیر بزرگتری به ما می‌دهد. از طرفی با افزایش مقدار آلفا تاثیر این مقادیر را می‌توان بیشتر کرد. پس با افزایش سایز پنجره و مقدار آلفا تصویر شارپ‌تری خواهیم داشت.

```
I= imread('Lena.bmp');
IR = rgb2gray(I);
blurred = IR;
blurred1 = IR;
blurred2 = IR;
blurred3 = IR;
for i = 0:1:50
   blurred = myFilterBox(blurred,3);
end
for i = 0:1:100
   blurred1 = myFilterBox(blurred1,3);
end
for i = 0:1:200
   blurred2 = myFilterBox(blurred2,3);
end
for i = 0:1:300
   blurred3 = myFilterBox(blurred3,3);
end
subplot(4,1,1),
imshow(blurred,[]),title;('50')
subplot(4,1,2),
imshow(blurred1,[]),title;('100')
subplot(4,1,3),
imshow(blurred2,[]),title;('200')
subplot(4,1,4),
imshow(blurred3,[]),title('300');
```

```
function [c] =
myFilterBox(NIm,window)
 %Defining the box filter mask
w=(1/window*window)*ones(win
dow,window)    ;
[ma, na] = size(NIm);
[mb, nb] = size(w);

 %To do convolution
c = zeros( ma+mb-1, na+nb-1 );
for i = 1:mb
   for j = 1:nb
      r1 = i;
      r2 = r1 + ma - 1;
      c1 = j;
      c2 = c1 + na - 1;
      c(r1:r2,c1:c2) =
c(r1:r2,c1:c2) + w(i,j) *
double(NIm);
   end
end


 %extract region of size(a) from c
r1 = floor(mb/2) + 1;
r2 = r1 + ma - 1;
c1 = floor(nb/2) + 1;
c2 = c1 + na - 1;
c = c(r1:r2, c1:c2);

end
```

```
function [out] = medianFilter(im,N)
im_pad = padarray(im, [floor(N/2)
floor(N/2)]);
im_col = im2col(im_pad, [N N],
'sliding');
sorted_cols = sort(im_col, 1, 'ascend');
med_vector = sorted_cols(floor(N*N/2)
+ 1, :);
out = col2im(med_vector, [N N],
size(im_pad), 'sliding');
end
```

```
function verticalEdge(I,mask)
I=double(I);
In=I      ;
%Rotate image by 180 degree first flip
up to down then left to right
mask=flipud(mask)  ;
mask=fliplr(mask) ;
for i=2:size(I, 1)-1
   for j=2:size(I, 2)-1
%       multiplying mask value with the
corresponding image pixel value
       neighbour_matrix=mask.*In(i-
1:i+1, j-1:j+1)  ;

avg_value=sum(neighbour_matrix(:)) ;
       I(i, j)=avg_value ;
   end
end
figure, imshow(uint8(I));
end
```

```
function[out] =
unsharpMasking(img,alpha,filterSize)
blurred =
imgaussfilt(img,'FilterSize',filterSize);
out = img + (alpha*(blurred-img));
end
```

```
function [out] = averageFilter(an,b)
[m,n]=size(an);
z=ones(b);
[p,q]=size(z);
w=1:p;
x=round(median(w));
anz=zeros(m+2*(x-1),n+2*(x-1));
for i=x:(m+(x-1))
   for j=x:(n+(x-1))
      anz(i,j)=an(i-(x-1),j-(x-1));
   end
end
sum=0;
x=0;
y=0;
for i=1:m
   for j=1:n
      for k=1:p
         for l=1:q
            sum=
sum+anz(i+x,j+y)*z(k,l);
            y=y+1;
         end
         y=0;
         x=x+1;
      end
      x=0;
      out(i,j)=(1/(p*q))*(sum);
      sum=0;
   end
end
figure, imshow(uint8(out))
end
```

مراجع

- Digital Image Processing / Rafael C. Gonzalez, 4th Edition
- [1] http://www.cs.unc.edu/~lazebnik/spring10/lec05_filter.pdf adapthisteq#:~:text=imadjust%20increases%20the%20contrast%20of,histeq%20performs%20histogram%20equalization.&text=adapthisteq%20performs%20contrast%2Dlimited%20adaptive%20histogram%20equalization
- https://www.geeksforgeeks.org/matlab-edge-detection-of-an-image-without-using-in-built-function/
- https://stackoverflow.com/questions/27535535/matlab-median-filter-code

.