فیلترها

علیرضا لرستانی

اطلاعات گزارش

تاریخ: 1399/09/07

واژگان کلیدی:
فیلتر
فیلتر مکانی
نویز
نویز نمک فلفل
فیلتر میانه یا median
فیلتر میانگین یا average
فیلترهای خطی
فیلترهای غیر خطی
نویز گاوسی
Blur
sharp

چکیده

در پردازش تصویر ما به دنبال تکنیک هایی برای بهبود نقاط ضعف تصاویر هستیم. یکی از این تکنیک ها فیلترینک است که برای که به وسیله ی آن میتوان یک عکس را اصلاح کرد و کیفیت آن را افزایش داد. به عنوان مثال شما یک عکس را فیلتر میکنید که برخی از ویژگی های آن را مهم تر جلوه دهید یا برخی از ویژگی های آن را حذف کنید.

1-مقدمه

نوشتار حاضر، به بررسی و پیاده سازی (با زبان برنامه نویسی متلب) برخی از فیلتر های مهم و کاربردی و تاثیر آن ها بر روی عکس هایی با ویژگی های گوناگون میپردازد.
در این تمرین شیوه حوزه مکانی یا spatial domain مورد بررسی قرار میگیرد که به صورت زیر تعریف میشود:
"اعمال یک سری عملیات بر روی پیکسل های تصویر با توجه به شدت و موقعیت آن پیکسل در تصویر".

در حوزه مکانی 2 دسته عملیات یا فیلتر وجود دارند :
فیلترهای خطی که در آن همسایگی های یک پیکسل با وزن های مشخص باهم جمع می شوند و پس از نرمالایز شدن ، به عنوان مقدار پیکسل قرار داده می شوند.
و فیلترهای غیر خطی که در آن خروجی بر اساس مرتبه پیکسل های همسایه به دست می آید.
از فیلتر های مکانی برای حذف نویز ، شارپ کردن تصویر و... استفاده میشود.

* پست الکترونیک نویسنده مسئول:
alirezalorestani2010@gmail.com

1. دانشجو، دانشکده مهندسی، دانشگاه فردوسی مشهد

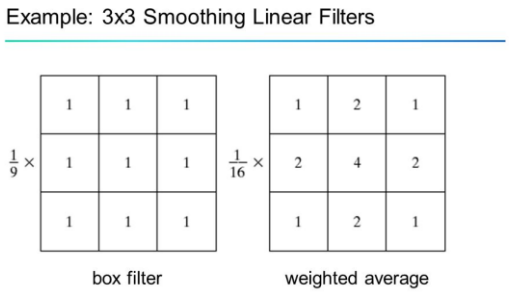
2-شرح تکنیکال

Box Filter : 3.1

در این تمرین درباره box filter صحبت خواهیم کرد:

در این فیلتر ابتدا پنجره ای انتخاب میکنیم. این پنجره میتواند سایز های مختلفی داشته باشد به طور مثال 3x3 و یا 10x10 و یا ...

این فیلتر به این صورت عمل میکند که مقدار هر پیکسل را با مقدار میانگین پیکسل هایی که درون پنجره قرار میگیرند (به صورت وزن دار یا غیر وزن دار) جایگزین میکند.



اثر ناخواسته این فیلتر blur شدن (تار شدگی) و از بین رفتن لبه هاست.

Median Filter :3.2

در این بخش ابتدا نویز نمک – فلفل و گوسین را به عکس اضافه خواهیم کرد سپس تلاش میکنیم نویز ها را به وسیله ی فیلتر میانه و جعبه حذف کنیم.

- فیلتر میانه :

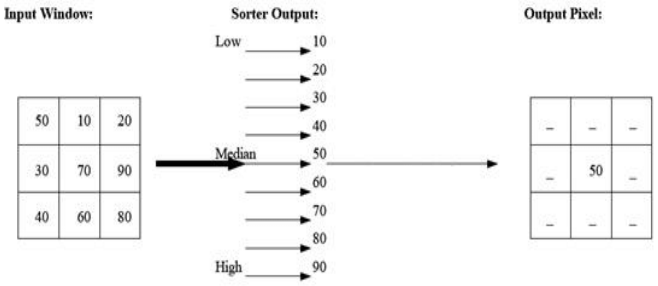
فیلتر میانه از انواع غیر خطی فیلتر هاست که به حذف نویز از تصاویر با حفظ لبه ها کمک میکند.

این فیلتر برای حذف نویز نمک – فلفل بسیار موثر عمل میکند.

در فیلتر میانه نیز ابتدا پنجره ای با سایز دلخواه در نظر میگیریم. سپس مقدار هر پیکسل با مقدار میانه پیکسل هایی که درون پنجره قرار میگیرند جایگزین میکند.

برای این کار ابتدا پیکسل ها را بر اساس مقدار آن ها مرتب میکند سپس داده وسطی یا همان میانه را پیدا میکند.

در زیر مثالی را با هم مشاهده میکنیم:



مقدار 70 با 50 که میانه پیکسل های درون پنجره است جایگزین میشود.

- نویز نمک – فلفل :

این نویز را اینگونه تعریف میکنیم:

" فقط برخی از پیکسل های تصویر دارای نویز هستند اما مقدار آن بسیار زیاد است (نزدیک به 0 یا 255)"

به مانند آنکه دانه های نمک و فلفل بر روی عکس پاشیده شده اند.

در زیر نمونه هایی از این نویز با مقادیر مختلف را مشاهده میکنید:

- نویز گوسین :

نویز گاوسی یک نویز استاتیک است که از تابع توزیع چگالی احتمال نرمال (تابع گاوسی) پیروی میکند؛ یعنی، مقادیر این نویز، توزیع گاوسی دارند. اگر از نزدیک و با دقت به

تصاویر روی فیلمهای گرفته شده با دوربینهای قدیمی تر یا تصاویر نوارهای ویدیویی که مدت طولانی آرشیو شده اند نگاه شود، ذرات ریز با الگوهای تصادفی قابل مشاهده است.

:3.3

Sharpening, Bluring and Noise Removal

در این بخش ابتدا با تلفن همراه خود عکسی از سوژه ای تاریک گرفتم که در ابتدا برای چشم انسان واضح نبود.
در ادامه با استفاده از تکنیک های گوناگون تلاش کردم تا آن را واضح تر کنم. در بخش شرح نتایج و نتیجه گیری مراحل کار به ترتیب توضیح داده شده است.

Edge Detection :3.4

در این بخش با به کارگیری متد هایی به دنبال یافتن لبه ها در تصاویر هستیم.
مجموعه ای از محاسبات ریاضی منجر به یافت نقاطی در تصویر میشود که در آن ها روشنایی تصویر به یکباره تغییر کرده، ما از این نقاط به عنوان لبه یاد میکنیم.
در واقع لبه های تصویر نقاطی هستند که پیکسل ها به یک باره تغییر مقدار میدهند و روال قبلی خود را ادامه نمیدهند.

Unsharp Masking :3.5

$$(1 - \alpha)I + \alpha I' = I + \alpha(I' - I),$$

در فرمول فوق I تصویر اصلی و I′ تصویر هموار شده است.
افزایش مقدار آلفا سبب افزایش تاثیر تصویر هموارشده و کاهش آن سبب کاهش تاثیر میشود.

- فیلتر گوسین:

فیلتری است که پاسخ پالس آن یک تابع گاوسی است.

3-شرح نتایج و نتیجه گیری

3.1.1

از مشکلات box filter میتوان به موراد زیر اشاره کرد:

- اگر در همسایگی پیکسل، پیکسلی وجود داشته باشد که نویز زیادی داشته باشد (یعنی مقدار آن تفاوت زیادی با مقادیر سایر پیکسل های زیر پنجره داشته باشد) روی میانگین تاثیر زیادی میگذارد.
- اگر پیکسل انتخابی روی لبه باشد پس از اعمال فیلتر مقدار آن تغییر خواهد کرد و باعث blur شدن آن پیکسل خواهد شد به طوریکه شاید آن لبه دیگر با چشم قابل تشخیص نباشد.
- تعداد محاسبات بالا از دیگر معضلات این روش است به طوریکه اگر سایز پنجره 100x100 باشد باید 10000 بار عمل جمع و ضرب انجام دهیم.

3.1.2

این فیلتر باعث blur شدن تصویر و از بین رفتن لبه ها میشود به طوریکه در دفعات تکرار زیاد ممکن است دیگر حتی تصویر قابل تشخیص نباشد.
در نتیجه اعمال مکرر این فیلتر نه تنها باعث بهبود عکس و از بین رفتن ویژگی های منفی آن نمیشود بلکه باعث محو شدن و از بین رفتن تصویر میشود.

3.1.3

در این بخش نتایج اعمال box filter با سایز پنجره 3x3 که کد آن را در بخش آخر نوشتار (پیوست) قرار خواهم داد را در دغعات مختلف مشاهده خواهیم کرد.

ابتدا تصویر اصلی را با هم مشاهده میکنیم:

سپس آن را به gray تبدیل میکنیم:

در تصویر زیر نتیجه یک بار اعمال box filter بر روی تصویر gray شده (تصویر سمت راست) در کنار تصویر اصلی gray شده (تصویر سمت چپ) را مشاهده میکنیم:

همچنین نتیجه ده بار اعمال box filter بر روی تصویر gray شده (تصویر سمت راست) در کنار تصویر اصلی gray شده (تصویر سمت چپ) :

نتیجه سی بار اعمال box filter بر روی تصویر gray شده (تصویر سمت راست) در کنار تصویر اصلی gray شده (تصویر سمت چپ) :


و در نهایت نتیجه صد بار اعمال box filter بر روی تصویر gray شده (تصویر سمت راست) در کنار تصویر اصلی gray شده (تصویر سمت چپ) به صورت زیر خواهد بود :


همانطور که در بخش قبل اشاره کردیم اعمال مکرر این فیلتر باعث blur شدن تصویر میشود.

3.2.1

ابتدا نویز نمک – فلفل با مقادیر density متفاوت را به عکس می افزاییم سپس سعی میکنیم نویز های ایجاد شده را با اعمال فیلتر میانه با سایز پنجره متفاوت رفع کنیم.
در انتها با استفاده از تابع immse مقادیر MSE را برای هر تصویر محاسبه میکنیم و نتیجه گیری نهایی خود را انجام میدهیم.

تصویر با نویز نمک – فلفل با مقدار density 0.05 و سایز پنجره 3x3 :


تصویر با نویز نمک – فلفل با مقدار density 0.05 و سایز پنجره 5x5 :


تصویر با نویز نمک – فلفل با مقدار density 0.05 و سایز پنجره 7x7 :


تصویر با نویز نمک – فلفل با مقدار density 0.05 و سایز پنجره 9x9 :

تصویر با نویز نمک – فلفل با مقدار 0.1 density و سایز پنجره 3x3 :

تصویر با نویز نمک – فلفل با مقدار 0.1 density و سایز پنجره 9x9 :

تصویر با نویز نمک – فلفل با مقدار 0.1 density و سایز پنجره 5x5 :

تصویر با نویز نمک – فلفل با مقدار 0.2 density و سایز پنجره 3x3 :

تصویر با نویز نمک – فلفل با مقدار 0.1 density و سایز پنجره 7x7 :

تصویر با نویز نمک – فلفل با مقدار 0.2 density و سایز پنجره 5x5 :

همانطور که مشاهده میشود هر چه تراکم یا density نویز کم تر باشد و همچنین سایز پنجره کوچک تر باشد فیلتر میانه موفق تر عممل خواهد کرد.

3.2.2

ابتدا نویز گوسین با مقادیر density متفاوت را به عکس می افزاییم سپس سعی میکنیم نویز های ایجاد شده را با اعمال فیلتر میانه و همچنین جعبه با سایز پنجره متفاوت رفع کنیم.

تصویر با نویز نمک – فلفل با مقدار density 0.2 و سایز پنجره 7x7 :

تصویر با نویز نمک – فلفل با مقدار density 0.2 و سایز پنجره 9x9 :

مقدار MSE برای هر تصویر در مقایسه با تصویر اصلی به شرح زیر است:

	3x3	5x5	7x7	9x9
0.05	69.77	106.44	140.77	173.61
0.1	88.04	132.55	172.76	211.85
0.2	127.14	178.68	229.22	294.03

تصویر با نویز گوسین با مقدار density 0.01 و سایز پنجره 3x3 و فیلتر میانه :

تصویر با نویز گوسین با مقدار density 0.01 و سایز پنجره 5x5 و فیلتر میانه :

تصویر با نویز گوسین با مقدار density 0.01 و سایز پنجره 7x7 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.01 density و سایز پنجره 9x9 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.05 density و سایز پنجره 7x7 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.05 density و سایز پنجره 3x3 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.05 density و سایز پنجره 9x9 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.05 density و سایز پنجره 5x5 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.1 density و سایز پنجره 3x3 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.1 density و سایز پنجره 5x5 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.01 density و سایز پنجره 3x3 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار 0.1 density و سایز پنجره 7x7 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.01 density و سایز پنجره 5x5 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار 0.1 density و سایز پنجره 9x9 و فیلتر میانه :


تصویر با نویز گوسین با مقدار 0.01 density و سایز پنجره 7x7 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار 0.01 density و سایز پنجره 9x9 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار 0.05 density و سایز پنجره 7x7 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار 0.05 density و سایز پنجره 3x3 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار 0.05 density و سایز پنجره 9x9 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار 0.05 density و سایز پنجره 5x5 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار 0.1 density و سایز پنجره 3x3 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار density 0.1 و سایز پنجره 5x5 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار density 0.1 و سایز پنجره 7x7 و فیلتر جعبه :


تصویر با نویز گوسین با مقدار density 0.1 و سایز پنجره 9x9 و فیلتر جعبه :


در انتها با استفاده از تابع immse مقادیر MSE را برای هر تصویر محاسبه میکنیم و نتیجه گیری نهایی خود را انجام میدهیم.

مقدار MSE با اعمال فیلتر میانه:

	3x3	5x5	7x7	9x9
0.01	120.29	138.86	167.00	195.38
0.05	275.35	289.50	317.40	346.69
0.1	764.21	772.77	802.54	834.95

مقدار MSE با اعمال فیلتر جعبه یا میانگین:

	3x3	5x5	7x7	9x9
0.01	122.15	131.12	172.82	219.19
0.05	272.80	277.01	315.00	358.47
0.1	743.85	745.03	774.64	815.23

باز هم مانند بخش قبلی به نظر میرسد نمونه هایی با کمترین سایز پنجره و کمترین تراکم نویز پس از اعمال هر دو فیلتر میانه و جعبه بهترین نتیجه را داشته اند.

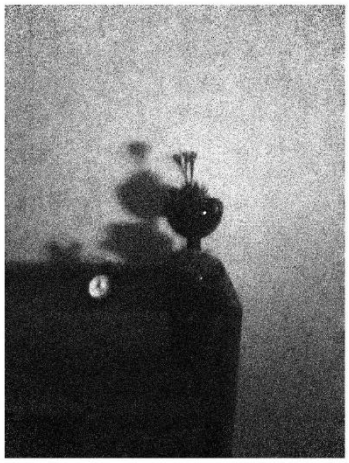
3.3

تصویر اولیه گرفته شده توسط دوربین تلفن همراه در نور کم:

ابتدا تصویر را سیاه سفید میکنیم:

سپس با استفاده از تابع imadjust کنتراست تصویر را افزایش میدهیم:

سپس با استفاده از تابع imsharpen تصویر را شارپ تر میکنیم:



همانطور که مشاهده میشود تصویر نویز زیادی دارد و نوی آن از نوع نمک – فلفل است. برای برطرف کردن این نویز ها از فیلتر میانه استفاده میکنیم.

تصویر تهایی پس از اعمال 10 مرتبه فیلتر میانه بر روی تصویر:

همانطور که مشاهده میشود تصویر در مقایسه با تصویر ابتدایی بسیار واضح تر شده است.

3.4.1

سه فیلتر داده شده را بر روی تصاویر اعمال میکنیم:

$$\frac{1}{2}\begin{bmatrix}1 & 0 & -1\end{bmatrix}$$

$$\frac{1}{6}\begin{bmatrix}1 & 0 & -1\\1 & 0 & -1\\1 & 0 & -1\end{bmatrix}$$

$$\frac{1}{8}\begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix}$$

فیلتر اول فقط دو پیکسل قبل و بعد را بررسی میکند بنابراین ممکن است لبه ای را تشخیص دهد که اصلا وجود ندارد.

فیلتر دوم و سوم پیکسل های مورب را نیز بررسی میکنند. در فیلتر دوم همه ی لبه ها چه قبل و بعد و چه مورب دارای یک اهمیت اند اما از دید فیلتر سوم لبه های قبل و بعد از لبه های مورب اهمیت بیشتری دارند.

اما همانطور که پیداست هیچ یک از این سه فیلتر بر روی تصویر موردنظر ما نتیجه ی مطلوبی را فراهم نمی آورند.

3.4.2

دو فیلتر رابرت را بر روی تصویر اعمال میکنیم:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

همانطور که از تصاویر پیداست فیلترهای پیشنهادی رابرت بهتر عمل میکنند و لبه ها بیشتر قابل تشخیص اند. در نتیجه اگر ما جزئیات لبه ها را بخواهیم فیلترهای پیشنهادی رابرت انتخاب بهتری از فیلترهای بخش قبل هستند و بالعکس.

4-پیوست

توابع ساخت نویز

```
imageWithSaltAndPepperNoise = imnoise(img,'salt & pepper',0.2);
imageWithGaussianNoise = imnoise(img,'gaussian',0.05);
```

تابع فیلتر جعبه

```
function boxFilter(originalImg,img,times,windowSize)

filteredImg = zeros(size(img),class(img));
temp = img;

for i=1 : times
    for k = 1 : size(temp,1)
        filteredImg(:,:,1) = conv2(temp(:,:,1),ones(windowSize,windowSize)/(windowSize^2),'same');
    end
    temp = filteredImg;
end

fprintf("MSE : %0.4f \n",immse(filteredImg,originalImg));
subplot(1,2,1),imshow(img);
subplot(1,2,2),imshow(filteredImg);
end
```

تابع فیلتر میانه

```
function medianFilter(img,noisyImg,windowSize)
a = double(noisyImg);
median = a;
[r,c] = size(a);
i = round(windowSize/2);
for x = 2:1:r-1
    for y = 2:1:c-1
        if(x-i> 0 && x + i<=512 && y-i>0 && y+i<=512)
            a1 = a((x-i:x+i),(y-i:y+i));
            a1 = reshape(a1 ,[1 size(a1,1)*size(a1,2)]);
            a2 = sort(a1);
            med = a2(round(length(a2)/2));
            median(x,y) = med;
        end
    end


end

fprintf("MSE : %0.4f \n",immse(uint8(median),img));

subplot(1,2,1),imshow(noisyImg);
subplot(1,2,2),imshow(uint8(median));
end
```

تابع اعمال فیلتر های تشخیص لبه

```
function edgeDetection(img)
filter1 = [1 0 -1]/2;
filter2 = [1 0 -1;1 0 -1;1 0 -1]/6;
filter3 = [1 0 -1;2 0 -2;1 0 -1]/8;
result1 = uint8(conv2(single(img),filter1,'same'));
result2 = uint8(conv2(single(img),filter2,'same'));
result3 = uint8(conv2(single(img),filter3,'same'));

filter4 = [1 0; 0 -1];
filter5 = [0 1; -1 0];
result4 = zeros(size(img),class(img));
result5 = zeros(size(img),class(img));
for k = 1 : size(img,1)

    result4(:,:,1) = conv2(img(:,:,1),filter4,'same');
    result5(:,:,1) = conv2(img(:,:,1),filter5,'same');
end
% imshow(result1);
% imshow(result2);
% imshow(result3);
% imshow(result4);
% imshow(result5);

end
```