به نام خدا

دانشگاه تهران

دانشکده‌ی مهندسی برق و کامپیوتر

پردازش سیگنال‌های زمان‌گسسته

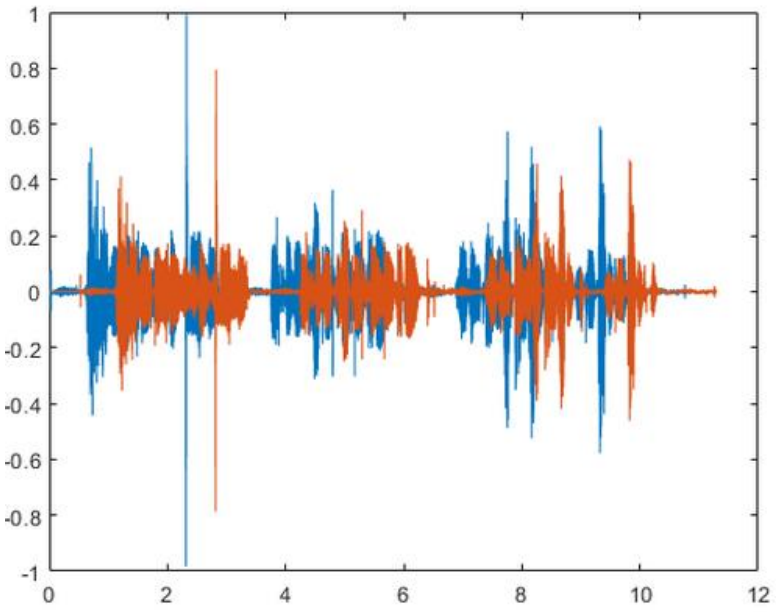
گزارش پروژه 3

نام و نام خانوادگی	محمد عبائیانی
شماره‌ی دانشجویی	810198432

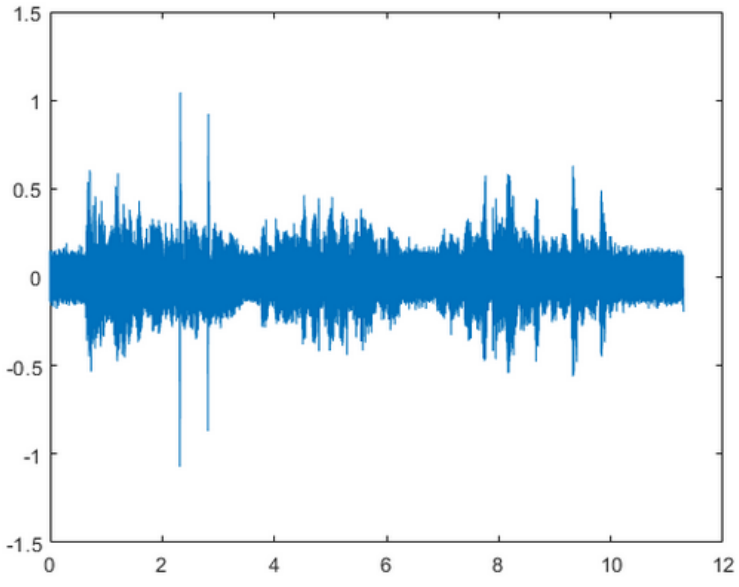
بخش اول

عملیات اضافه کردن نویز به صورت زیر است:

ابتدا سیگنال را با شیفت یافته یخودش جمع کرده تا اکوایجاد شود: (مقدار بتا برابر 0.5 است)

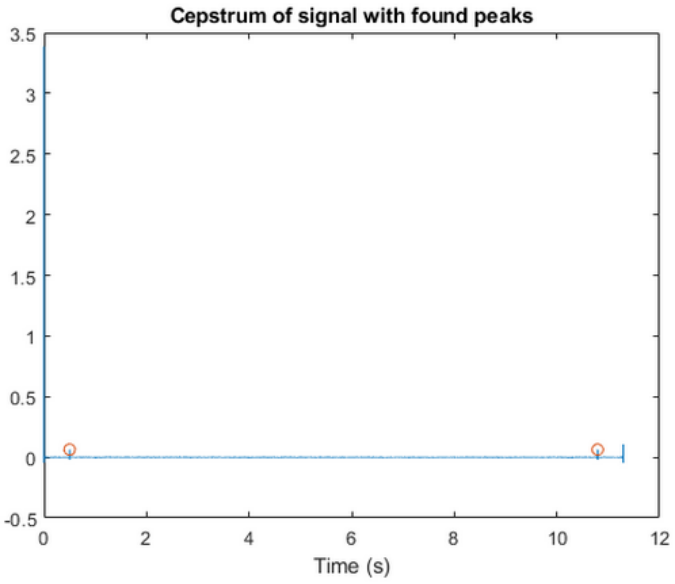


سپس نویز اضافه میشود:



حذف اکو

در این بخش ابتدا نمودار کپستروم را میکشیم و قله های میانی را به دست می آوریم.

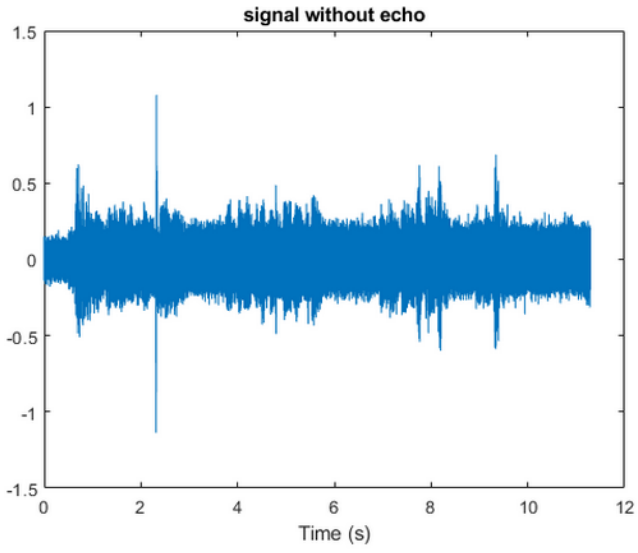


پیک اول برابر B+1 میباشد بنابر این بتا را میتوان به دست آورد که با مقدار اصلی یکیست:

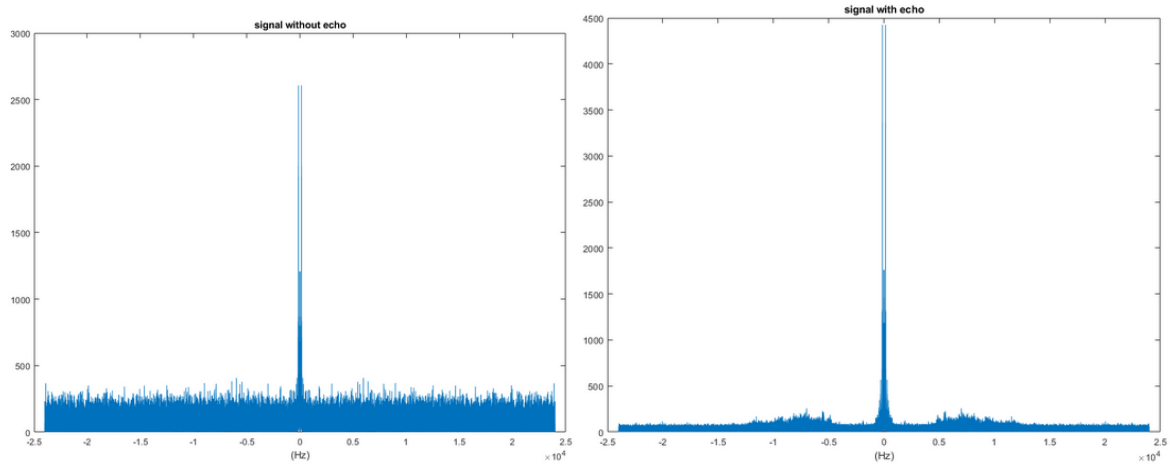
```
function y = rmecho(x,alpha,fs)
c = rceps(x);

[px,locs] = findpeaks(c,'Threshold',0.01,'MinPeakDistance',0.02);
dl = locs(1)-1;
fprintf("Beta is %f" , dl/fs )
y = filter(1,[1 ,zeros(1,dl-1), alpha],x);
audiowrite("noisy_voice.wav",y,fs)

end
```

```
Beta is 0.500000
```

با اعمال تابع filterو دانستن مقدار Bسیگنال را فیلتر میکنیم:



در حوزه ی فرکانس به صورت زیر است:

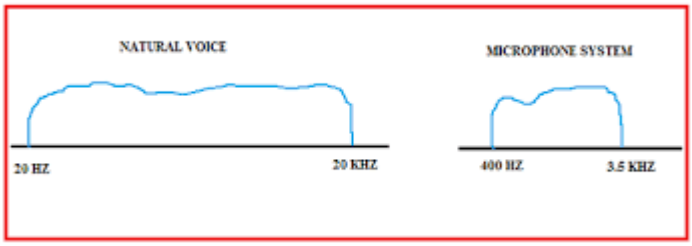


از آنجایی که تاخیر تنها اختلاف فاز را به همراه دارد سیگنال اکو دار در دامنه تنها اسکیل یافته ی سیگنال اصلی میباشد(نویز اسکیل پیدا نمیکند زیرا اکو نیافته)، بنابراین در سیگنال اکو دار فرکانس های سیگنال اصلی نسبت به نویز نمود بیشتر دارند(SNRبیشتر است) اما شکل اندازه ی سیگنال یکسان است.

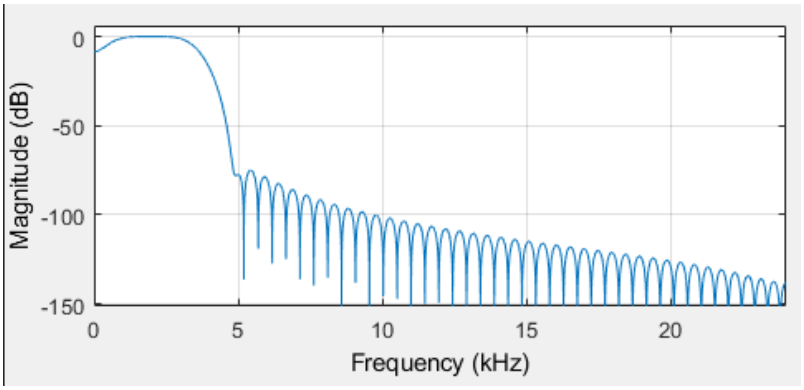
حذف نویز

این بخش را با طراحی یک فیلتر blackmanمرتبه 100 انجام میدهیم.

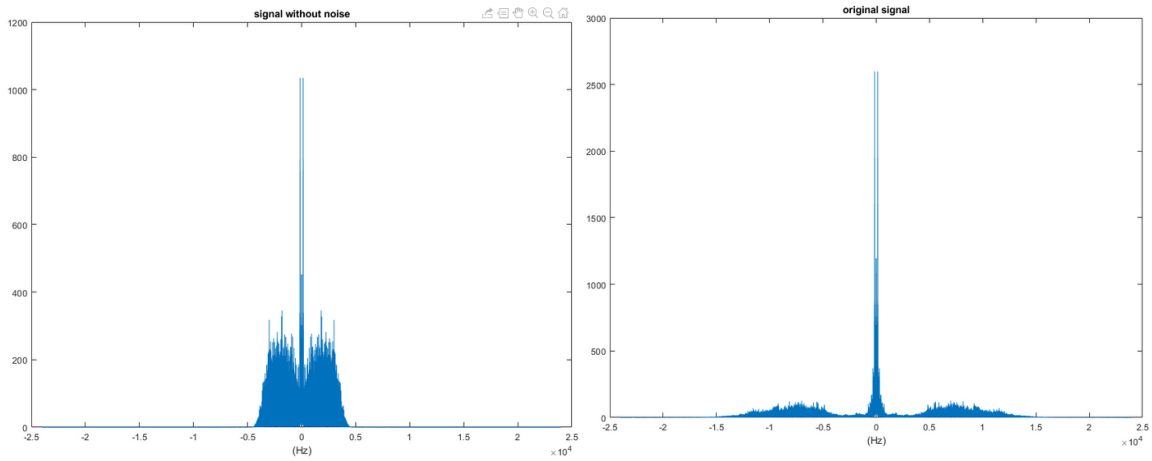
برای فرکانس های باند طبق عکس زیر عمل میکنیم.(400 و 3500)



پاسخ فیلتر به شکل زیر است:

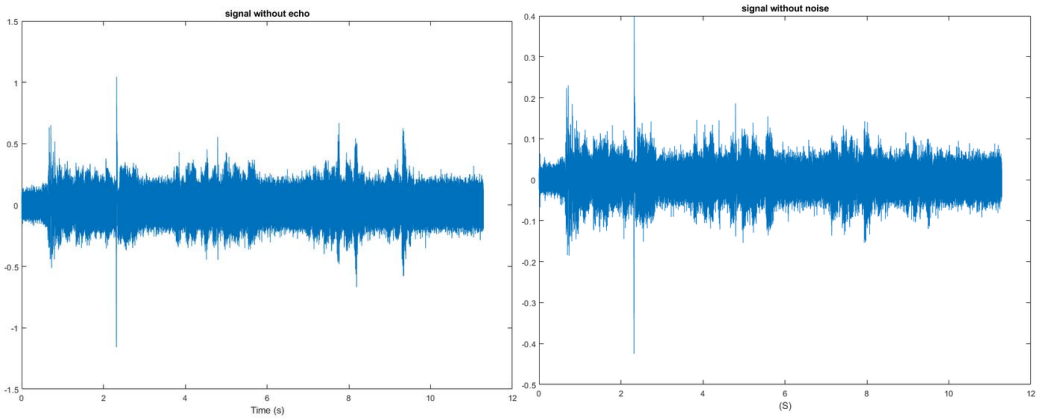


در خروجی مشاهده میشود که نویز کاهش یافته اما همچنان مقداری باقی میماند، زیرا در فرکانس فیلتر شده همچنان نویزی با ماهیت تصادفی وجود دارد که از بین نمیرود.

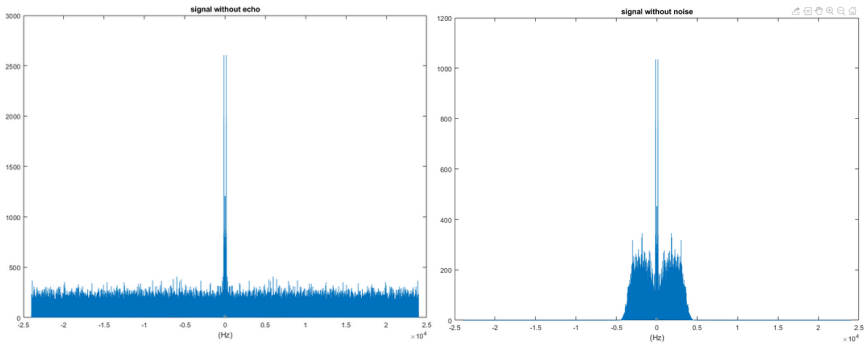


همانطور که مشاهده میشود قسمتی از سیگنال نیز حدود 5 کیلوهرتز تا 15 کیلوهرتز حذف شده که موجب کاهش نسبی کیفیت صدا شده است، اما قسمت اصلی صدا به طور کامل از فیلتر عبور میکند که بخشی از نویز را به همراه دارد.

مقایسه با سیگنال نویز دار:



در حوزه ی زمان صوت اصلی مقداری از نویز متمایز و قابل تشخیص شده، اما سیگنال همچنان مقداری نویز را در فرکانس های عبور داده شده همراه دارد



بخش دوم: پردازش تصویر

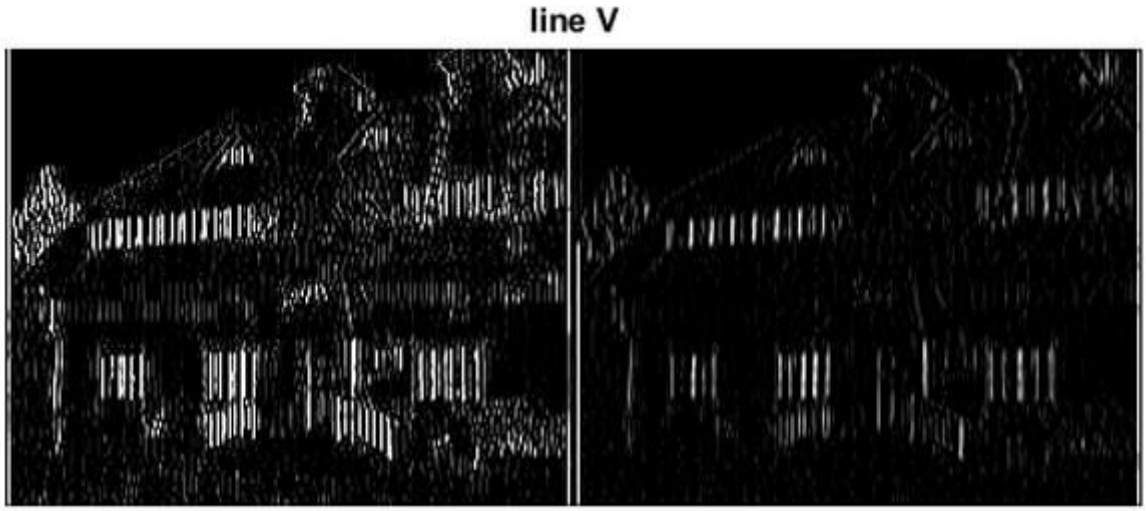
1. کرنل ها

کرنل Identity

این کرنل تغییری روی عکس ایجاد نمیکند و به do_nothing kernel معروف است.

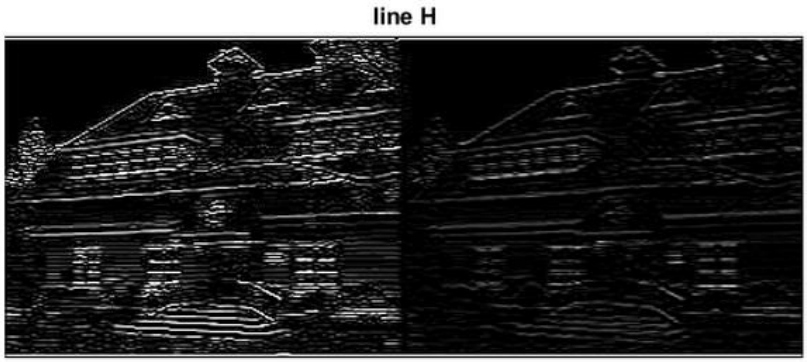
کرنل Line V

این کرنل خط های عمودی را در عکس مشخص کرده و باقی قسمت ها را صفر میکند.



کرنل Line H

این کرنل خط های افقی را مشخص کرده و مابقی عکس را صفر میکند.

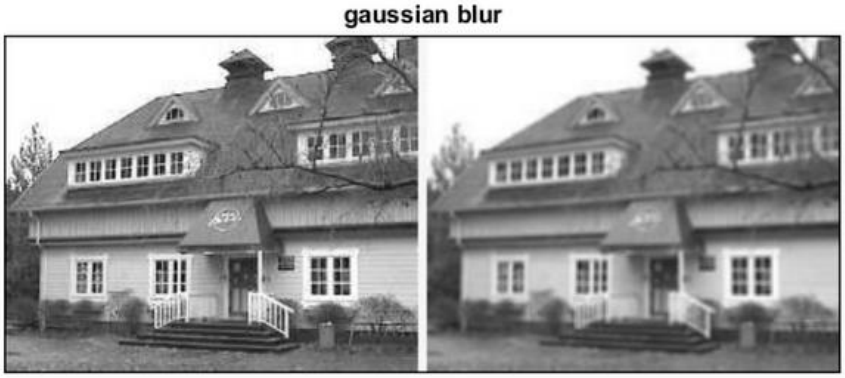


کرنل moving average

این کرنل مقدار هر نقطه را توسط میانگین 98 خانه ی کناری (و خود خانه) تخمین میزند.

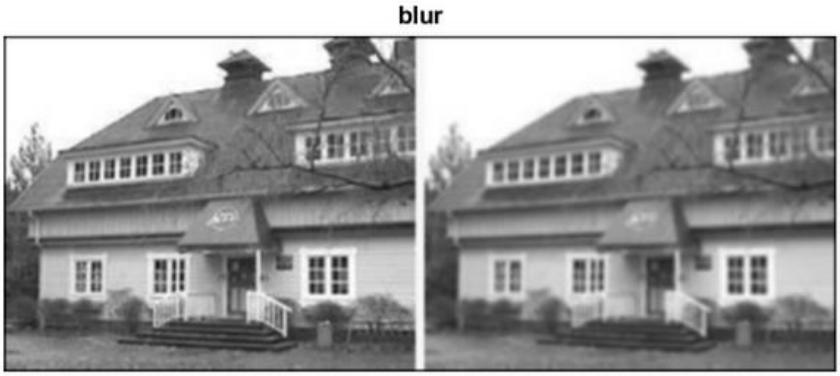
کرنل gaussian

این کرنل هر خانه را با توزیع گاوسی 9 خانه ی کناری(وزن دادن به هر خانه با ضرایب نرمال) به دست می آورد.



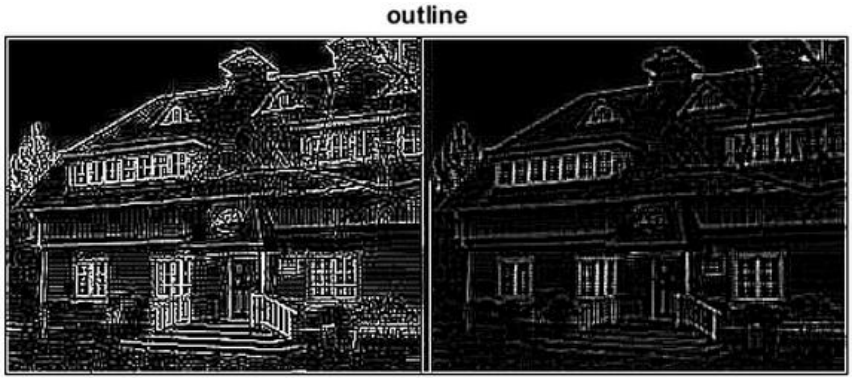
کرنل Blur

این کرنل با مانند کرنل گاوسی عمل کرده تنها با این تفاوت که وزن خانه های کناری بیشتر بوده بنابراین تصویر بیشتر مات خواهد شد.



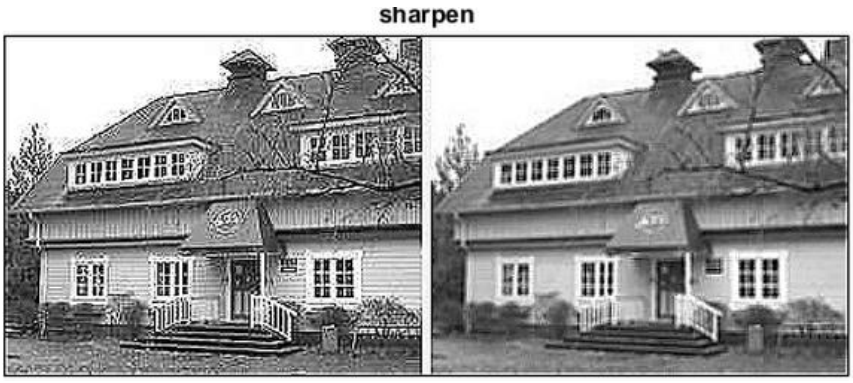
کرنل Outline

این کرنل خط های عکس را(عمودی و افقی) مشخص کرده و مابقی عکس را صفر میکند.



کرنل sharpen

این کرنل کنتراست عکس را بالا میبرد، یعنی تفاوت هر خانه با خانه های مجاورش را بیشتر میکند.



درونیابی تصویر

تصویر پس قبل و پس از اسکیل کردن :

در کوپک کردن تصویر مشکلی ایجاد نمیشود زیرا تمام دیتای مورد نظر وجود دارد، اما در بزرگنمایی درواقع up sampling رخ میدهد که باعث میشود دیتایی در بین دیتا های تصویر اصلی قرار بگیرد که مقدار آن در تصویر اصلی وجود ندارد، بنابراین این دیتا با نزدیک ترن دیتای اطراف خود پرشده که باعث پیکسلی شدن تصویر میشود.

پس از درونیابی خروجی ها به شکل زیر میباشند:

خروجی پس از 50 بار کانوالو کردن با کرنل گاوسی و 7 بار sharpen (به دلیل مات شدن)

عکس سمت چپ بدون sharpenو عکس سمت راست با sharpen

درونیابی با moving avg

در این حالت تصویر زودتر مات میشود

پس از 20 بار moving avg و 10 بار sharpen

عکس سمت چپ بدون sharpenو عکس سمت راست با sharpen

بخش امتیازی:

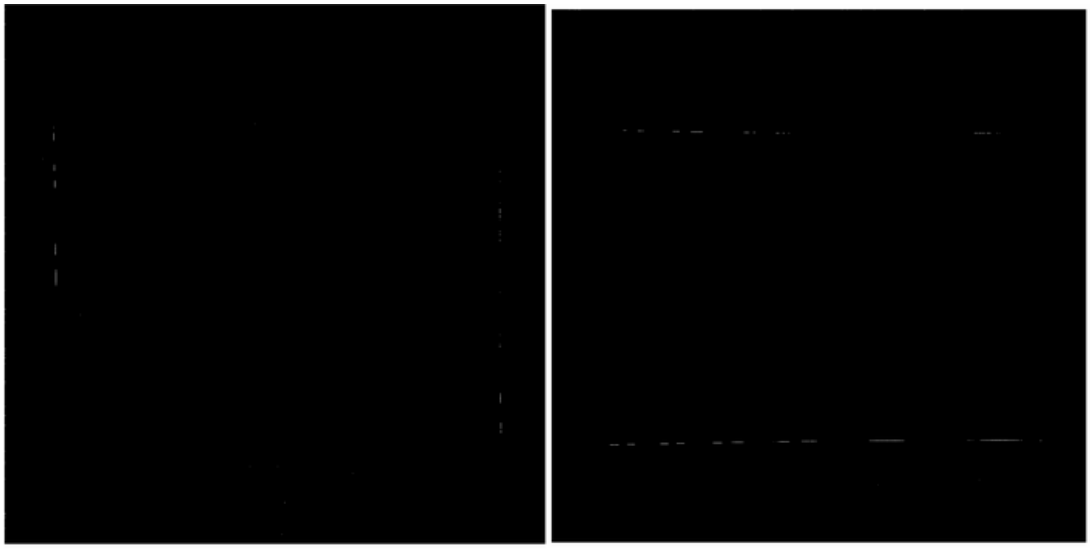
در این بخش الگوریتم به صورت زیر است.

هدف این است که با خروجی H line ضلع بالایی و پایینی و با خروجی V Line ضلع چپ و راست را به دست بیاوریم.

برای اینکه نویز تصویر کمتر شود و تنها نقاط اضلاع باقی بمانند نیاز است که چند مرحله فیلتر انجام شود.

در مرحله ی اول نقاطی که مقدارشان از 150 کمتر است را صفر میکنیم، سپس اندیس نقاط سفید را با دستور find درون آرایه ای ذخیره میکنیم.

خروجی ها به شکل زیر خواهد بود



در خروجی این قسمت همچنان نقاط سفیدی جز حاشیه ی کاغذ وجود دارد که 2 نوع هستند.

1. حاشیه ی کل تصویر
2. نویز های باقیمانده که به صورت نقاط کوچکی لا به لای تصویر وجود دارد.

برای قسمت اول حاشیه ی تصویر را 50 واحد در نظر گرفته و مقادیر حاشیه را حذف میکنیم.

```
[hl_row,hl_col] = find(Hlines);
hl_row = sort(hl_row);
hl_row = hl_row(hl_row >50 & hl_row<length(pic(:,1)) - 50);
```

برای قسمت دوم تنها خانه هایی را نگه میداریم که اختلافشان با خانه های مجاور صفر است. این عمل را 4 بار انجام میدهیم تا نویز های بیشت زا یک نقطه نیز حذف شوند(از آنجایی که نقاط زلع ها تعدادشان زیاد است حذف شدن چند مقدار از آنها تاثیر چندانی نخواهد داشت.

```
for i = 0:3
    hl_row = hl_row(diff(hl_row) == 0 );
end
```

در این مرحله نقاط موجود در آرایه تنها شامل ضلع ها میباشند، بنابراین با در نظر گرفتن حاشیه ی کمتر از 100 پیکسل میانگین مقادیر کمتر از این حاشیه(اندیس اول ماتریس +100) را به عنوان ضلع بالایی و میانگین مقادیر بیشتر این حاشیه به عنوان ضلع پایینی اعلام میکنیم.

```
top = round(mean(hl_row(hl_row<hl_row(1)+ 100)))
down = round(mean(hl_row(hl_row>hl_row(1)+ 100)))
```

این عمل را برای خروجی V lines نیز تکرار کرده و مقادیر ضلع چپ و راست را به دست می آوریم.

در نهایت با داشتن اندیس تمامی اضلاع میتوان مستطیل را رسم نمود.

خروجی: