

بسم الله الرحمن الرحيم

پروژه نهایی درس سیگنال ها و سیستم ها

سید محمد مهدی رضوی \_ ۹۷۵۲۲۱۵۷

در ابتدای کار تصاویر دانشکده ، تصویر خودم و دوستم و همچنین تصویر یک منظره زیبا را به فرم خاکستری درآوردم تا بتوانم انواع الگوریتم های یافتن لبه و همچنین حذف و اضافه کردن نویز را اعمال کنم.

```
computer = imread('IUST.jpg');
mahdi = imread('Mahdi.jpg');
nature = imread('Nature.jpg');

% figure
% imshow(computer);
% figure
% imshow(mahdi);
% figure
% imshow(nature);

G1 = rgb2gray(computer);
G2 = rgb2gray(mahdi);
G3 = rgb2gray(nature);

figure
imshowpair(G1 , computer , 'montage' );
title('IUST Grey & RGB Form');
figure
imshowpair(G2 , mahdi , 'montage' );
title('Mahdi Grey & RGB Form');
```

```
figure  
imshowpair(G3 , nature , 'montage' );  
title('Nature Grey & RGB Form');
```

IUST Grey & RGB Form



Mahdi Grey & RGB Form



Nature Grey & RGB Form



اولین الگوریتم سوبل خواهد بود که کد مطلب آن را نیز در زیر آورده ام. به نسبت دیگر الگوریتم ها ضعیف تر عمل می کند و لبه های کمتری را نمایش می دهد

با استفاده از تقریب سوبل و مشتق مرتبه اول را در نقاطی که شیب تصویر ماقزیم است پیدا می کند..

```
%Edge detection
FB1 = (0.2989 * double(computer(:,:,:1)) + 0.5870 *
double(computer(:,:,:2)) + 0.1140 * double(computer(:,:,:3)))/255;
FB2 = (0.2989 * double(mahdi(:,:,:1)) + 0.5870 *
double(mahdi(:,:,:2)) + 0.1140 * double(mahdi(:,:,:3)))/255;
FB3 = (0.2989 * double(nature(:,:,:1)) + 0.5870 *
double(nature(:,:,:2)) + 0.1140 * double(nature(:,:,:3)))/255;

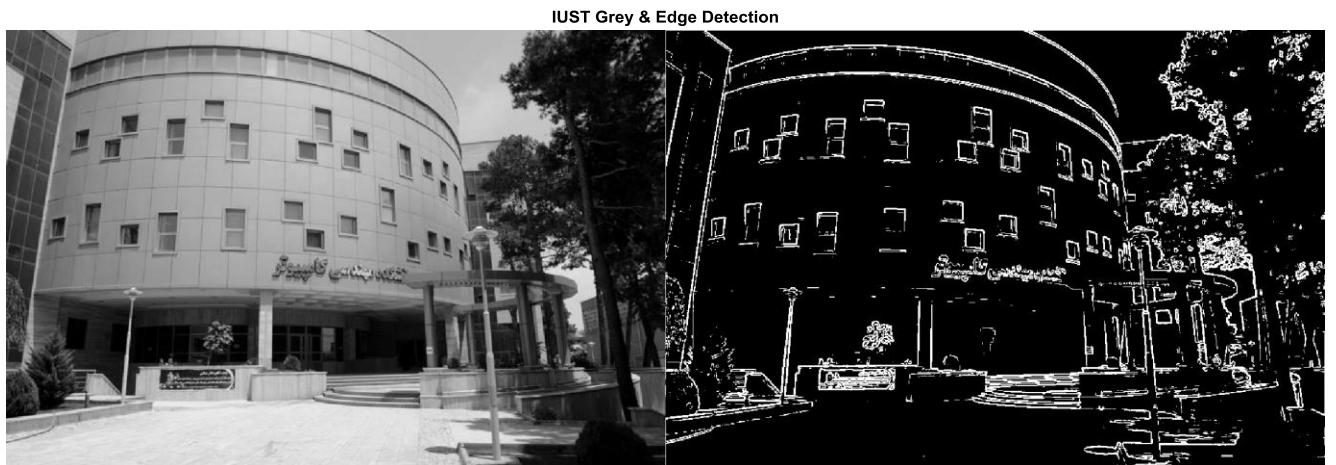
edgeFB1 = sobel_mex(FB1, 0.7);
Edge_FB1 = repmat(edgeFB1, [1 1 3]);
figure
imshowpair(FB1 , Edge_FB1 , 'montage');
title('IUST Grey & Edge Detection');
edgeFB2 = sobel_mex(FB2, 0.7);
```

```

Edge_FB2 = repmat(edgeFB2, [1 1 3]);
figure
imshowpair(FB2 , Edge_FB2 , 'montage');
title('mahdi Grey & Edge Detection');

edgeFB3 = sobel_mex(FB3, 0.7);
Edge_FB3 = repmat(edgeFB3, [1 1 3]);
figure
imshowpair(FB3 , Edge_FB3 , 'montage');
title('nature Grey & Edge Detection');

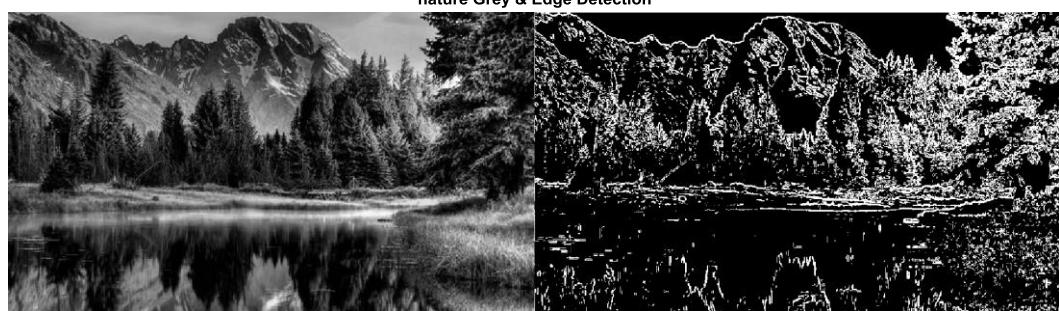
```



mahdi Grey & Edge Detection



nature Grey & Edge Detection



توضیحات مربوط به الگوریتم های یافتن لبه را در صفحه بعد مشاهده خواهید کرد.

Method	Description
'Sobel'	Finds edges at those points where the gradient of the image $I$ is maximum, using the Sobel approximation to the derivative.
'Prewitt'	Finds edges at those points where the gradient of $I$ is maximum, using the Prewitt approximation to the derivative.
'Roberts'	Finds edges at those points where the gradient of $I$ is maximum, using the Roberts approximation to the derivative.
'log'	Finds edges by looking for zero-crossings after filtering $I$ with a Laplacian of Gaussian (LoG) filter.
'zerocross'	Finds edges by looking for zero-crossings after filtering $I$ with a filter that you specify, $h$ .
'Canny'	Finds edges by looking for local maxima of the gradient of $I$ . The edge function calculates the gradient using the derivative of a Gaussian filter. This method uses two thresholds to detect strong and weak edges, including weak edges in the output if they are connected to strong edges. By using two thresholds, the Canny method is less likely than the other methods to be fooled by noise, and more likely to detect true weak edges.
'approxcanny'	Finds edges using an approximate version of the Canny edge detection algorithm that provides faster execution time at the expense of less precise detection. Floating point images are expected to be normalized in the range [0 1].

## Canny Method

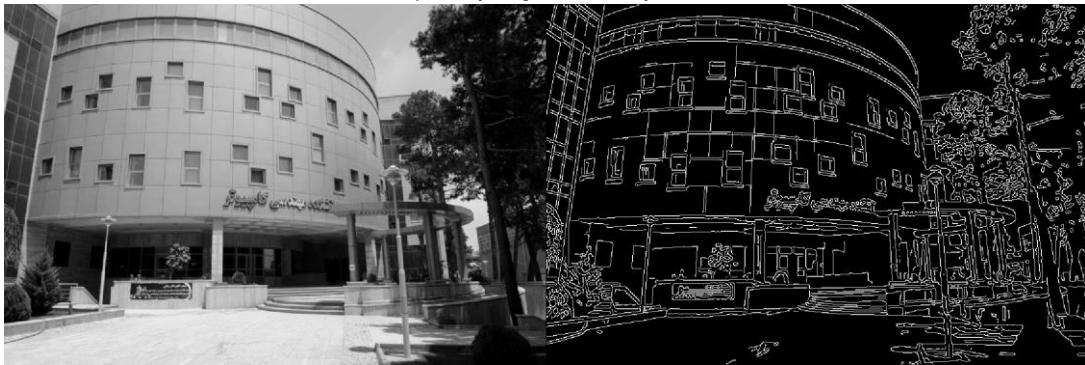
با جستجوی حداکثرهای محلی از شیب  $I$ . لبه ها را پیدا می کند. تابع `edge` با استفاده از مشتق یک فیلتر گاوی شیب را محاسبه می کند. در این روش از دو آستانه برای تشخیص لبه های قوی و ضعیف استفاده می شود ، از جمله لبه های ضعیف در خروجی اگر به لبه های قوی متصل شوند. با استفاده از دو آستانه ، روش `Canny` نسبت به سایر روش ها فریب صدا را می خورد و احتمالاً لبه های ضعیف واقعی را تشخیص می دهد. همانطور که در نمونه ها مشاهده میکنید الگوریتم بسیار قویتر است و تعداد بیشتری لبه را در عکس ها کشف کرده است.

```
BW1 = edge(FB1 , 'Canny');
figure
imshowpair(FB1 , BW1 , 'montage');
title('computer Grey & Edge Detection Canny Method');

BW2 = edge(FB2 , 'Canny');
figure
imshowpair(FB2 , BW2 , 'montage');
title('mahdi Grey & Edge Detection Canny Method');

BW3 = edge(FB3 , 'Canny');
figure
imshowpair(FB3 , BW3 , 'montage');
title('nature Grey & Edge Detection Canny Method');
```

computer Grey & Edge Detection Canny Method



mahdi Grey & Edge Detection Canny Method



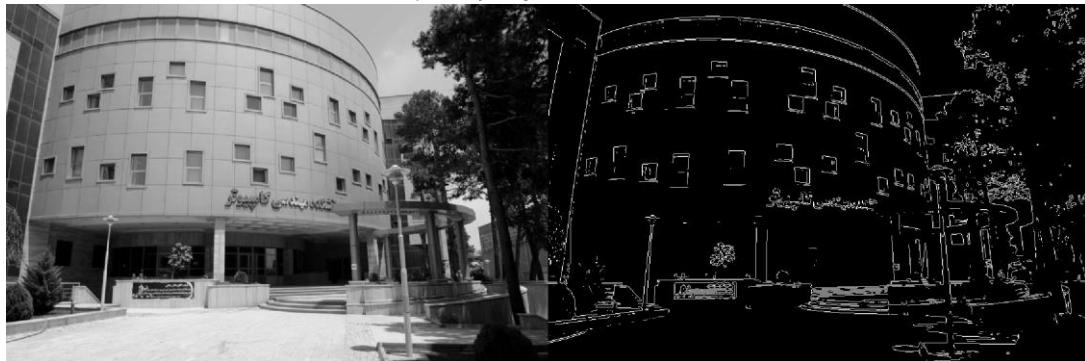
nature Grey & Edge Detection Canny Method



## Prewitt Method

این الگوریتم بسیار ضعیف تر عمل کرده و تعداد کمتری از لبه ها را پیدا کرده است.

computer Grey & Edge Detection Prewitt Method



mahdi Grey & Edge Detection Prewitt Method



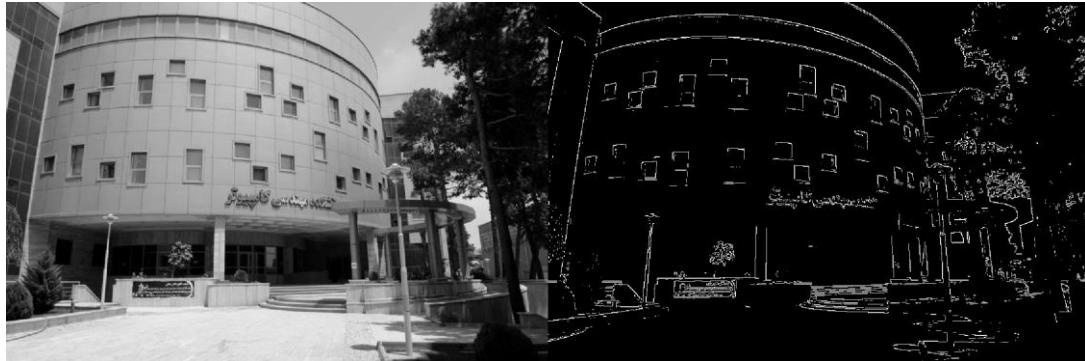
nature Grey & Edge Detection Prewitt Method



## Roberts Method

این الگوریتم نقاطی را یافته است که سطح تمایز بین دو سطح بسیار بالا بوده است و به نظرم برای تمایز میان سطوح فیزیکی بسیار مناسب است.

computer Grey & Edge Detection Roberts Method



**mahdi Grey & Edge Detection Roberts Method**



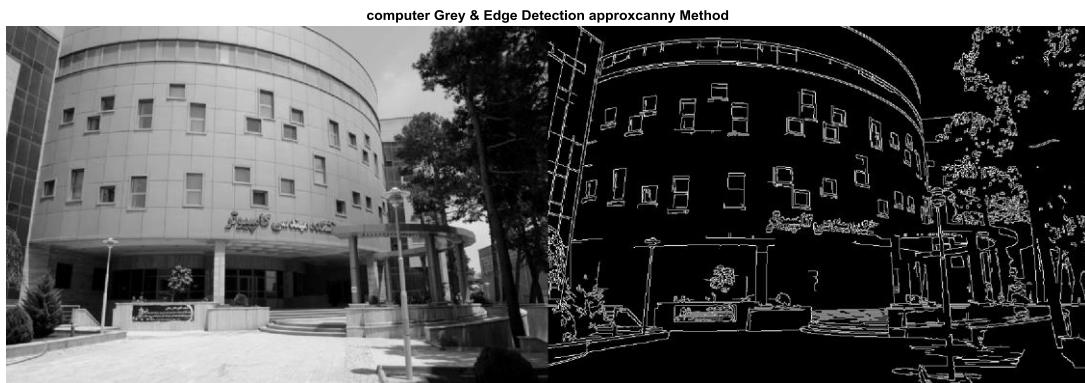
**nature Grey & Edge Detection Roberts Method**



## Approxcanny method

با استفاده از یک نسخه تقریبی از الگوریتم تشخیص لبه Canny ، لبه ها را پیدا می کند که زمان اجرای سریعتر را با هزینه تشخیص دقیق کمتر فراهم می کند. انتظار می رود تصاویر نقطه شناور در محدوده [۱۰] نرمال شوند.

همانطور که مشاهده میکنید لبه های ریز تر (حتی مو های سر من و دوستم) نیز کشف شده است.

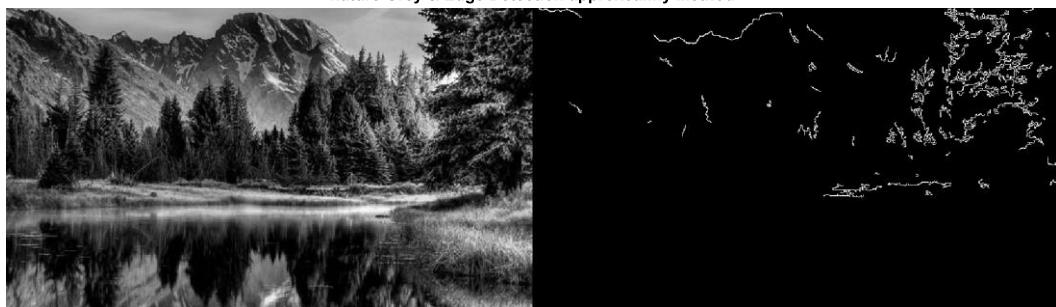


**mahdi Grey & Edge Detection approxcanny Method**



S

**nature Grey & Edge Detection approxcanny Method**



## Zerocross method

هر دوی الگوریتم های زیر برای یافتن لبه های کوچک خواهند بود و بسیار شبیه الگوریتم قبل هستند.

computer Grey & Edge Detection zerocross Method



mahdi Grey & Edge Detection zerocross Method



nature Grey & Edge Detection zerocross Method



## Log Method

computer Grey & Edge Detection log Method



mahdi Grey & Edge Detection log Method



nature Grey & Edge Detection log Method



```

Add & Remove noise from image
J1 = imnoise(G1 , 'gaussian',0,0.025);

title('Portion of the Image with ');

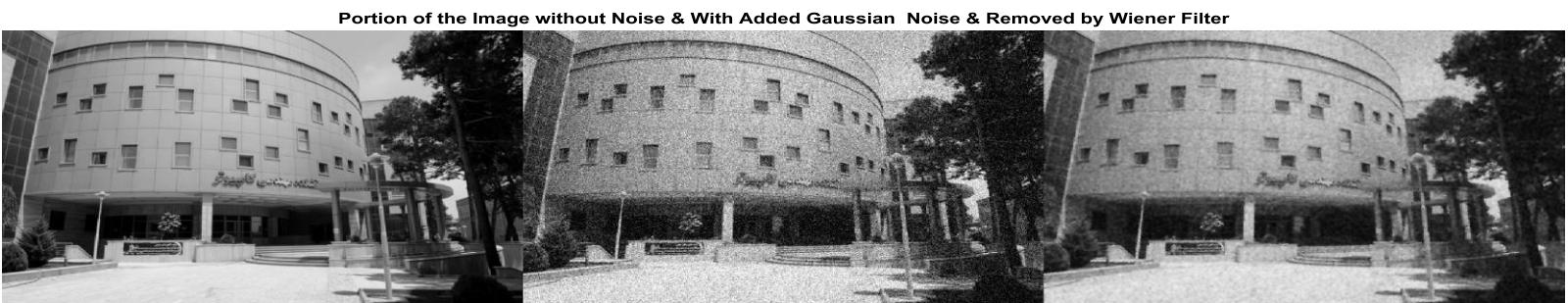
K = wiener2(J1 ,[5 5]);
multi = cat(2 , G1 , J1 , K);
montage(multi)

title('Portion of the Image without Noise & With Added Gaussian
Noise & Removed by Wiener Filter');

```

ابتدا از روش گوسین و با دو دامنه مختلف برای نویز های خود شکل های زیر را خواهیم داشت.  
 (از توزیع نرمال در این الگوریتم استفاده شده است)  
 بیشتر تصویر به حالت برفکی تر و مات تر خواهد بود.

دامنه نویز: ۰..۲۵

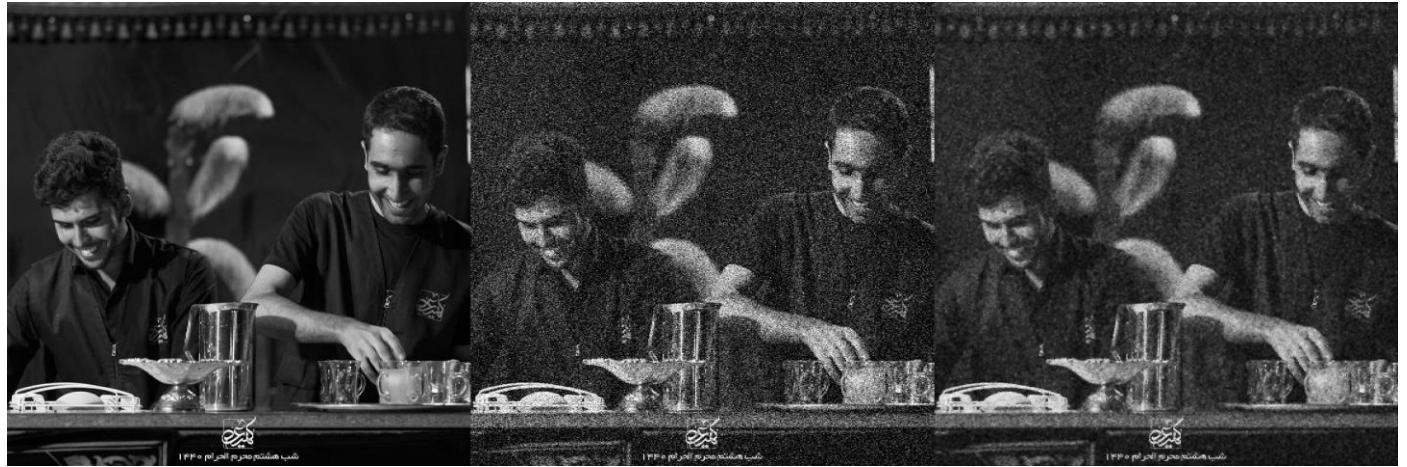


دامنه نویز : ۵۰۰

Portion of the Image without Noise & With Added Gaussian Noise & Removed by Wiener Filter



Portion of the Image without Noise & With Added Gaussian Noise & Removed by Wiener Filter



Portion of the Image without Noise & With Added Gaussian Noise & Removed by Wiener Filter



## Salt and PePPER Noise

در این روش با استفاده از چند نقطه رندوم نویز را اعمال خواهیم کرد. همانطور که مشاهده میکنید با رفع نویزهای موجود باز به شکل اولیه نخواهیم رسید و با کمی ارور همراه خواهد بود.

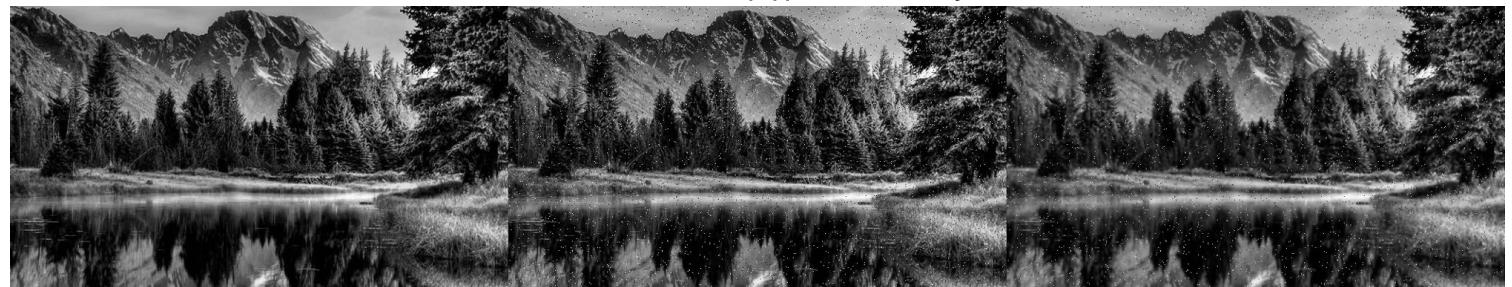
Without Noise & With Added Salt pepper & Removed by Wiener Filter



Without Noise & With Added Salt pepper & Removed by Wiener Filter



**Without Noise & With Added Salt pepper & Removed by Wiener Filter**



## Kmedian Method

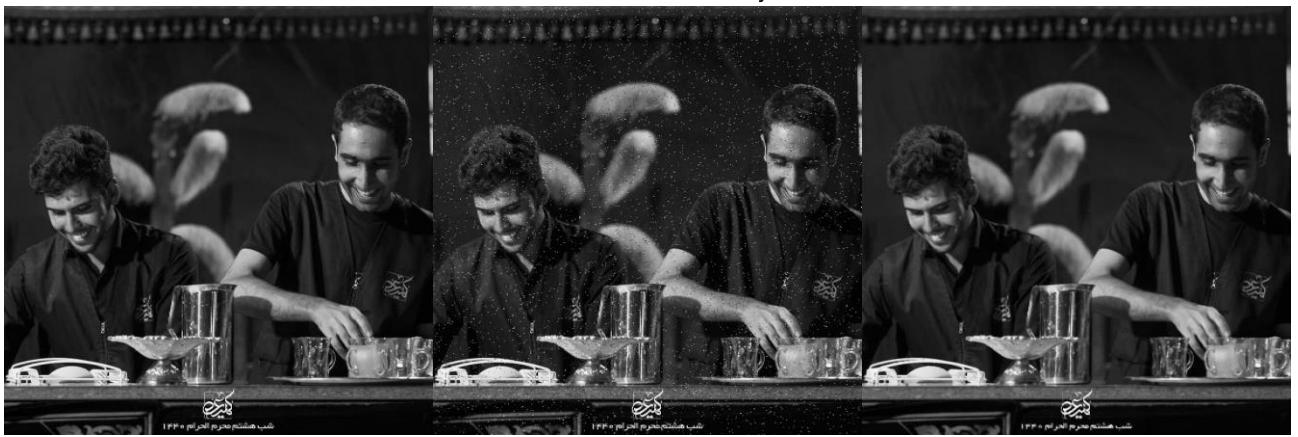
در انتها نیز از این clustering می باشد برای افزودن و از بین بردن نویز استفاده خواهیم کرد  
متد که بر اساس

باز هم تصویر رفع نویز شده با تصویر اولیه یکسان نخواهد بود.

Without Noise & With Kmedian & Removed by Kmedian Filter



Without Noise & With Kmedian & Removed by Kmedian Filter



**Without Noise & With Kmedian & Removed by Kmedian Filter**

