

Görüntü İşleme

Bilecik Şeyh Edebali Üniversitesi

Matlab Görüntü İşleme Filtreleme Projesi

Açıklama:

Matlab GUI kullanarak bir resim dosyasına uygulanan
filtrelemeler sonucu çıktının gözlenip yazdırılması

16700330224

Efecan Altay

2016

Yrd. Doç. Dr.

Ümit Çiğdem Turhal

Giriş

Görüntü İşleme Konusundan biri olan Filtreleme işlemlerinin Uzamsal Domainde Filtreleme örneklerini Matlab GUI arayüzüyle birlikte bir resmin girişi sonucu filtrelenmiş bir şekilde gösterilmesi ve yazdırılması programının çalışma prensibini ele almaktadır.

Projenin her bir yapım aşaması,kodların ne işe yaradıkları anlatılmaktadır fakat matlab fonksiyonların içeriklerini fazla ayrıntılı anlatılmamakla,fonksiyonların ne işe yaradıkları ve tanımları programın yardım bölümünden soru işaretlerine(?) tıklanarak öğrenilebilir.

Proje raporu 4.12.2016 tarihiyle başlamış olup geliştirmeler sonucu ek özellikler eklenecektir.

İçindekiler

Giriş.....	1
Matlab.....	3
Matlab Nedir ?	3
Matlabda Kod Nasıl Yazılır ?.....	3
Görüntü İşleme	4
Görüntü İşleme Nedir ?.....	4
Görüntüde Filtreleme	4
Filtre ve Filtreleme	4
Görüntüdeki Filtre Türleri	4
Averange Filtresi(Ortalama Filtre).....	4
Gaussian Filtresi(Ağırlıklı Ortalama Filtre)	5
Medyan Filtresi(Ortanca değer Filtresi)	5
Unsharp Filtresi	6
Motion Filtresi	6
Blurring Filtresi	6
Matlab GUI Kullanımı.....	7
Matlab Gui Nedir ?	7
GUIDE Editörünün Açılması.....	7
Eleman Ekleme ve Callback Fonksiyonu	8
Matlabta Kullanılan Filtreleme Fonksiyonları	9
Yumuşatma	9
Averange Filtresi-(imfilter Fonksiyonu).....	9
İmgaussfilt Fonksiyonu.....	9
Keskinleştirme	10
UnSharp Filtresi.....	10
Kenar Bulma	11
Log Filtresi	11
Canny Filtresi.....	11
Roberts Filtresi	12
Prewitt Filtresi	12
Sobel Filtresi	13
Laplace Filtresi.....	13
Kabartma	14
Gürültü Giderme	15
Gürültü Ekleme (İmnoise fonksiyonu)	15
Medyan Filtresi(medFilt).....	15

Matlab

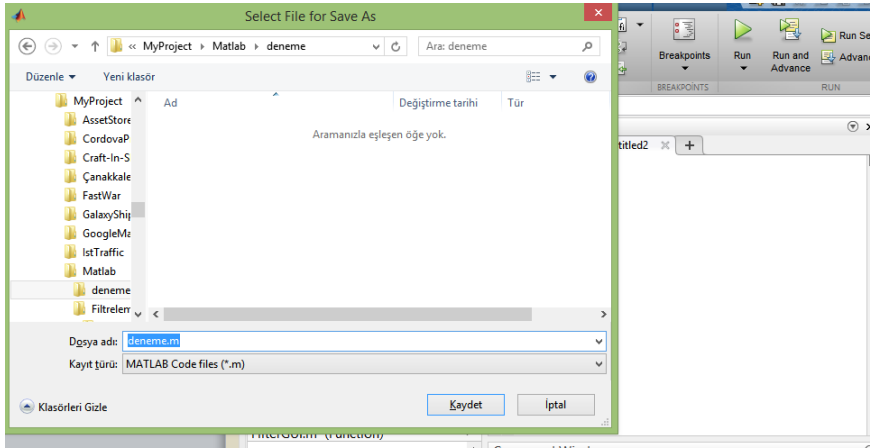
Matlab Nedir ?

Matlab bir çok verileri işlemeye ,göstermeye yardımcı programdır.Bilimsel alanda çok kullanılan,matematiksel alanda programlamaya yatkın, programlama dili kolay olan ve bir çok alanda program kütüphanesi yazılıp örnekleri yardım yoluyla gösterilen programdır.

Matlabın alt konularında birçok bilim alanında yazılmış kod kütüphaneleri bulunmaktadır.bu sayede hazır fonksiyonlarla matlab sayesinde birçok program geliştirilebilir.



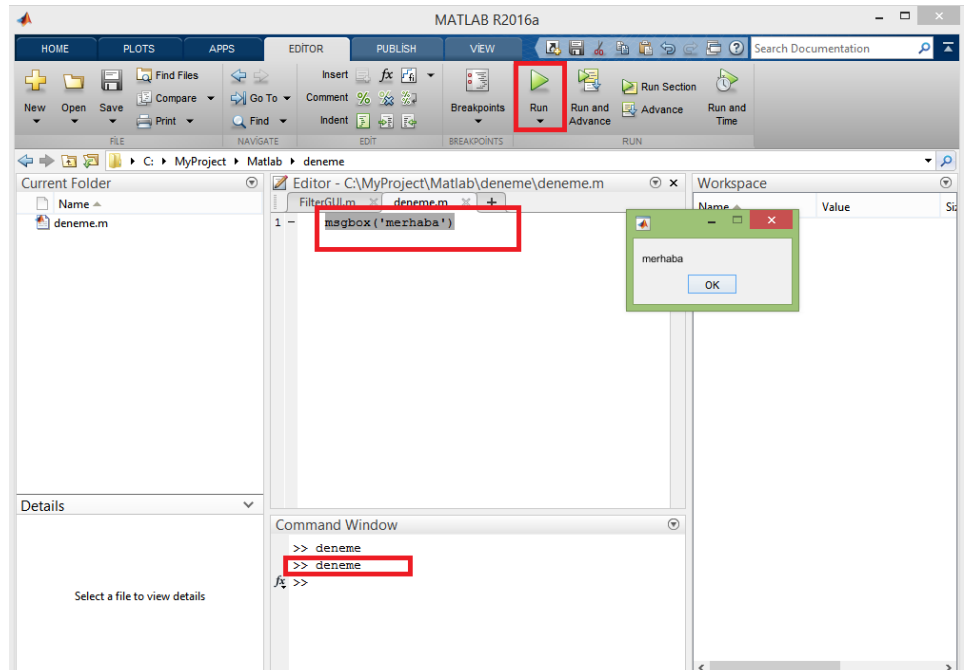
Matlabda Kod Nasıl Yazılır ?



Kurulu bir Matlab Programını açtığımızda karşımıza gelen Command Window olacaktır. **Command Window** yazdığımız kodları satır satır çalıştıran alt programdır. Buraya kodlarımızın bir satırını yazıp teker teker çalıştırabiliriz.fakat biz burada çalışmayacağız. Matlab dosyası oluşturup kodlarımızı oraya yazıp derleyip çalıştırmamız gerekecektir.

Bunun için **Ctrl+N** ile yeni dosya oluşturabiliriz. Ve **Ctrl+S** ile adını ve nereye kaydetmek istiyorsak orayı seçip kaydedebiliriz.

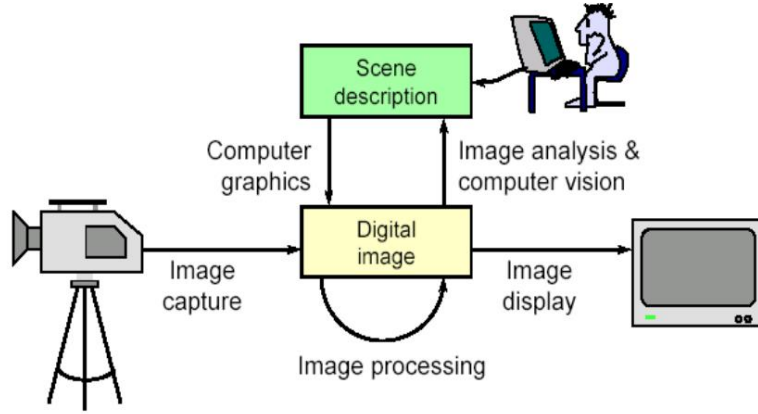
Bu işlemler sonucu karşımıza bir editör penceresinde kaydettiğimiz dosya görünecektir. Artık buraya işlem yapmak için matlab kodlarımızı yazıp EDITÖR panelindeki Run butonuna tıklayarak derleyip çalıştırabiliriz.Not: programın gördüğü proje klasörünü değiştirmemizi isteyebilir.Change folder yaparak işleme devam edebiliriz.



Görüntü İşleme

Görüntü İşleme Nedir ?

Bilgi teknolojilerinde yaşamımızdaki sinyalleri bilgisayarda işleyip ortaya sonuç çıkartmak için sensörler yardımıyla dış ortamdan verileri bilgisayara sayısal olarak aktarırız ve işleriz. Görüntü işleme ise görüntü verilerini bilgisayarda işlemesinde yardımcı bilim dalıdır. Görüntü işleme görüntü verileri alınıp birçok işlemden geçirilip ortaya



sonuç çıkarır. Filtreleme görüntü işlemenin yaptığı işlemlerin sadece biridir.

Görüntüde Filtreleme

Filtre ve Filtreleme

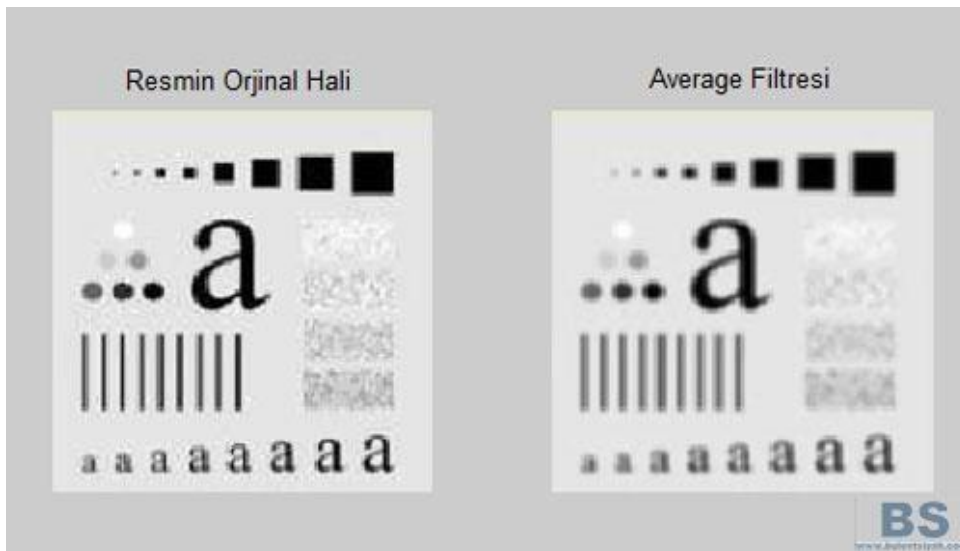
Verileri süzme işlemini gerçekleştiren yapıya filtre denir. Filtrede verilerin süzülme işlemine filtreleme denir. Filtremiz bir girişin verilerini farklı işlemler sonucu yeni bir çıkış elde eder.

Filtreleme Görüntüde Uzamsal Domainde filtreleme ve Frekans Domaininde filtreleme işlemleri olarak ikiye ayrılır. Uzamsal domainde filtreleme bir resmin zaman düzlemindeki anlık görüntüsü alınarak işlenmesidir. Frekans domaininde filtreleme ise uzamsal domainde alınan görüntünün zaman düzleminde frekans düzlemine dönüşümü sonucu frekans düzleminde oluşan görüntünün filtrelenmesi ve tekrar zaman düzlemine dönüştürülme işlemidir. İki farklı yol ile farklı sonuçlara ulaşılabilir.

Görüntüdeki Filtre Türleri

Average Filtresi (Ortalama Filtre)

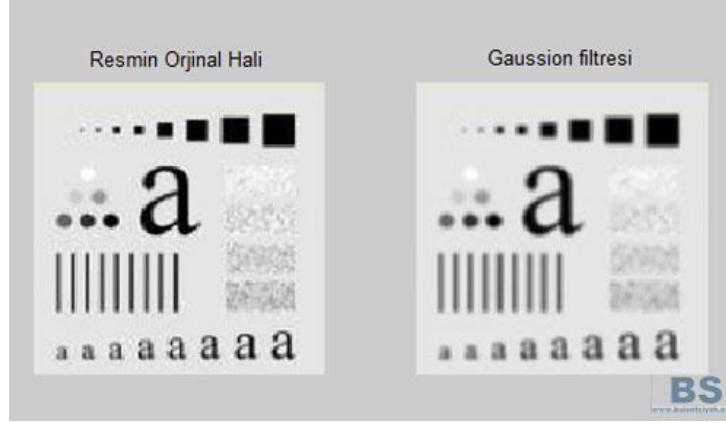
Resimdeki her piksel için piksel komşuları ile beraber ortalaması alınarak yeni bir resim



oluşturulur.sonuç olarak resim bulanıklaştırılmış olur.

Gaussian Filtresi(Ağırlıklı Ortalama Filtre)

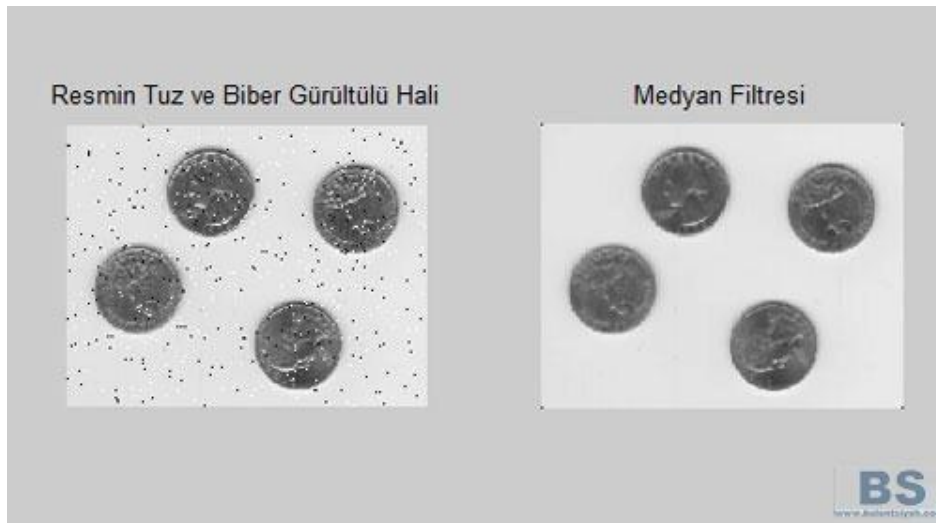
Ortalama filtrenin Gaussian dağılımını kullanarak biraz daha değiştirilmiş hali Gaussian filtresi olarak bilinir. Gaussian filtreleme aynı zamanda bir fourier dönüşümüdür. Gauss filtre ile sonsuz bir transfer fonksiyonuna karşılık mekansal alanda sonlu bir pencerede (tarama penceresi) filtreleme yapılabilmektedir. Bu da filtrelemenin temel problemini daha kolay çözülebilir hale getirir.



Medyan Filtresi(Ortanca değer Filtresi)

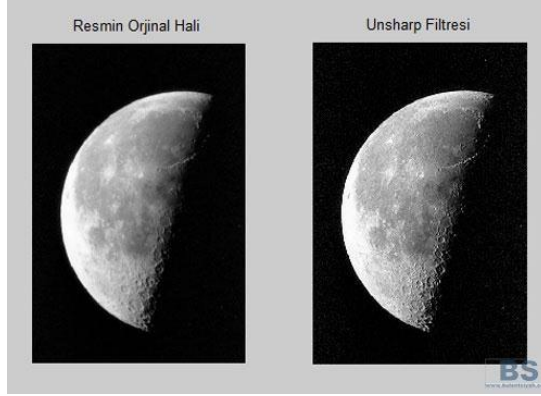
Bu filtreleme yönteminde, orjinal sıralanmış piksel komşularının arasındaki ortanca değer ile değiştirilir. Bunun ağırlıklı ortalama filtrelerinden farkı şudur: Ağırlıklı ortalama filtrelerinde, komşuların ağırlıklı ortalaması alınır, hesaplanan bu değer orijinal piksel ile yeniden ortalanarak sonuç bulunur. Ortanca filtresinde ise, komşuluk değerleri önce sıraya konulur, sonra ortadaki değer alınır. Bu değer doğrudan sonuç kabul edilir. Ortanca değeri net elde edebilmek için genellikle tek sayıda komşu seçilir. Eğer hesaplamada çift sayıda komşu kullanılırsa, bu durumda ortada kalan iki pikselin aritmetik ortalaması kullanılır.

Ortanca filtre; Uzaysal çözünürlüğü bozmadan, kopuk (bağımsız) nokta veya çizgi gürültülerini temizlemek için kullanışlıdır. Bu nedenle ikili (binary) gürültülerde başarılı olmasına rağmen Gaussian gürültüsünde kullanışlı değildir. Gürültü piksellerinin sayısı komşu piksellerin yarısına eşit veya daha fazla ise bu filtre pek başarılı çalışmaz.



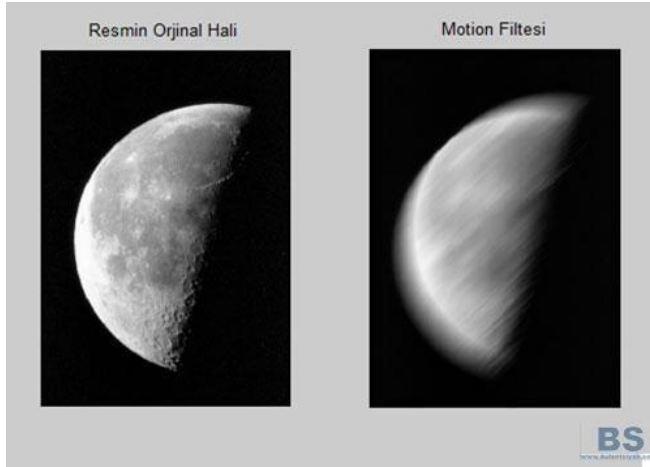
Unsharp Filtresi

Resimdeki ayrıntıları, keskin geçişleri belirginleştirmek, bulanıklaştırılmış görüntülerdeki ayrıntıları yeniden ortaya çıkarmak için kullanılır. Endüstriyel ve askeri alanda, tıbbi çalışmalarda ve diğer birçok alanda yararlıdır.



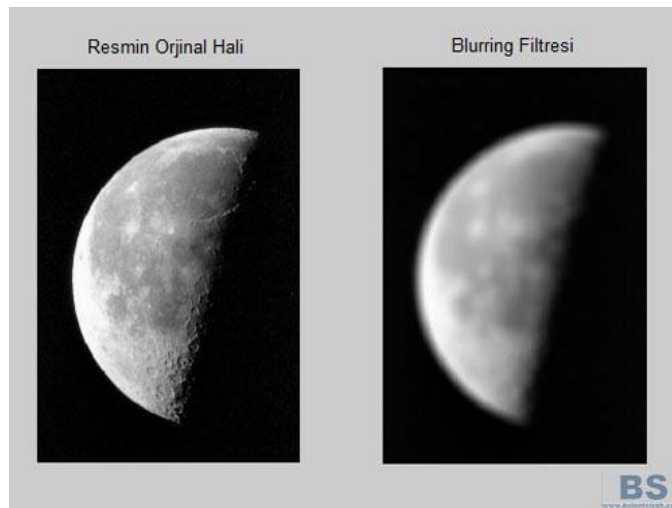
Motion Filtresi

Görüntüyü hareket esnasında çekilmiş gibi algılanmasına sebep olur.



Blurring Filtresi

Görüntüyü bulanıklaştırır.



Matlab GUI Kullanımı

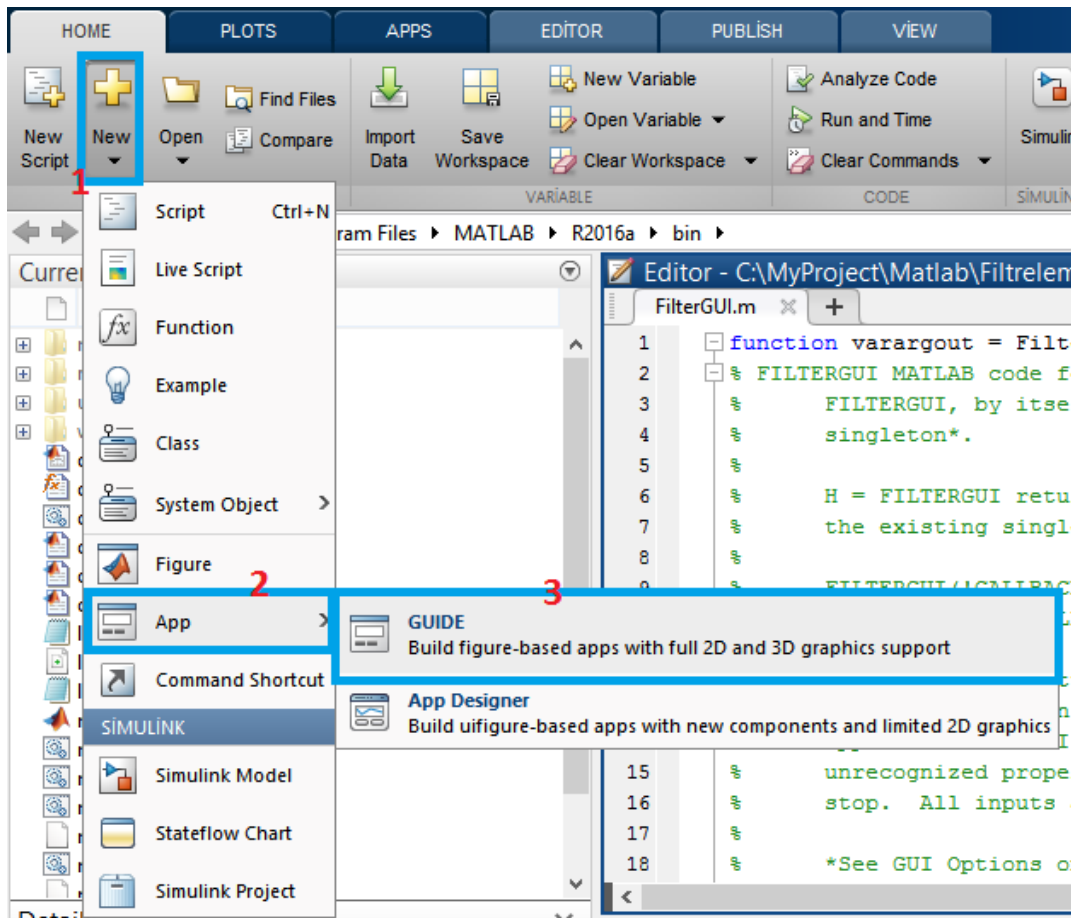
Matlab Gui Nedir ?

Graphical User Interface grafiksel kullanıcı arayüzü anlamına gelen gui,bilgisayar programlarının kullanıcılar ile iletişimini sağlamak için kullanılan görsel lerdir.Matlabtaki bu görsellere matlab gui adı verilir.

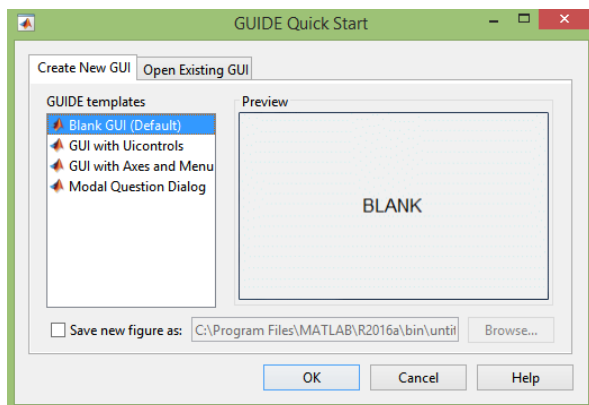
Matlabta yazdığımız kodları çalıştırmak için matlab gui kullanırız.Guileri oluşturması çok basittir ve sadece sürükle bırak ile buton ,text gibi elemanları formumunuzu içine yerleştiririz.

GUIDE Editörünün Açılması

Matlabta Gui oluşturmak için;



Proje dosyamızın içindeyken New -> APP -> GUIDE seneceklerini seçerek,

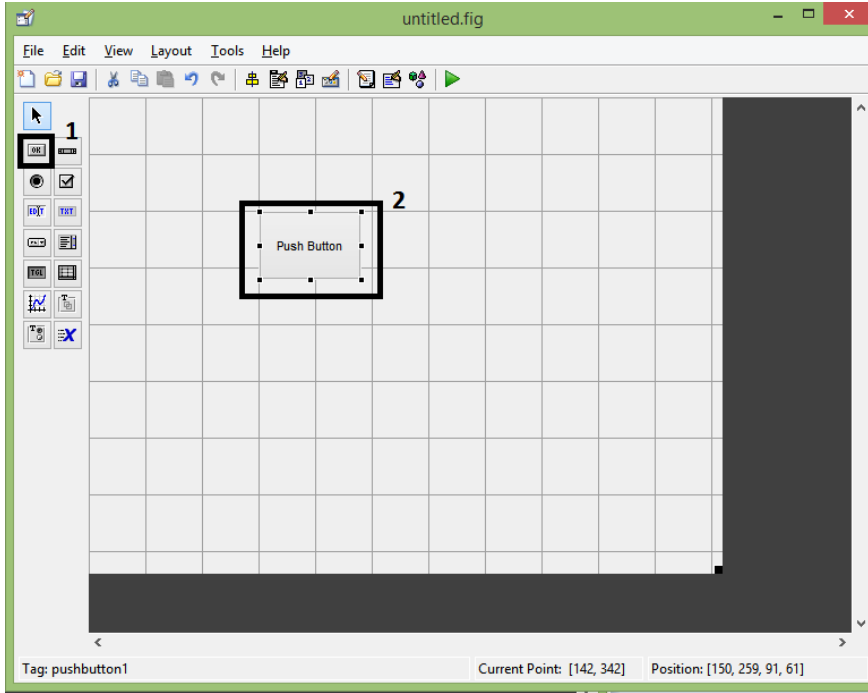


GUIDE Quick Start penceresiyle karşılaşıyoruz.burada projemiz için hazır başlangıç guileri mevcut bulunmaktadır.

Bize uygun şablonu seçip grafik arayüzümüzü tasarlamaya geçebiliriz.

Eleman Ekleme ve Callback Fonksiyonu

Yukardaki işlemlerden sonra editör karşımıza gelmektedir.



Şekil 1.


Şekil1de ki gibi 1. Aşamada elemanınız seçer ve 2 deki gibi çizdirme işlemini gerçekleştirir.

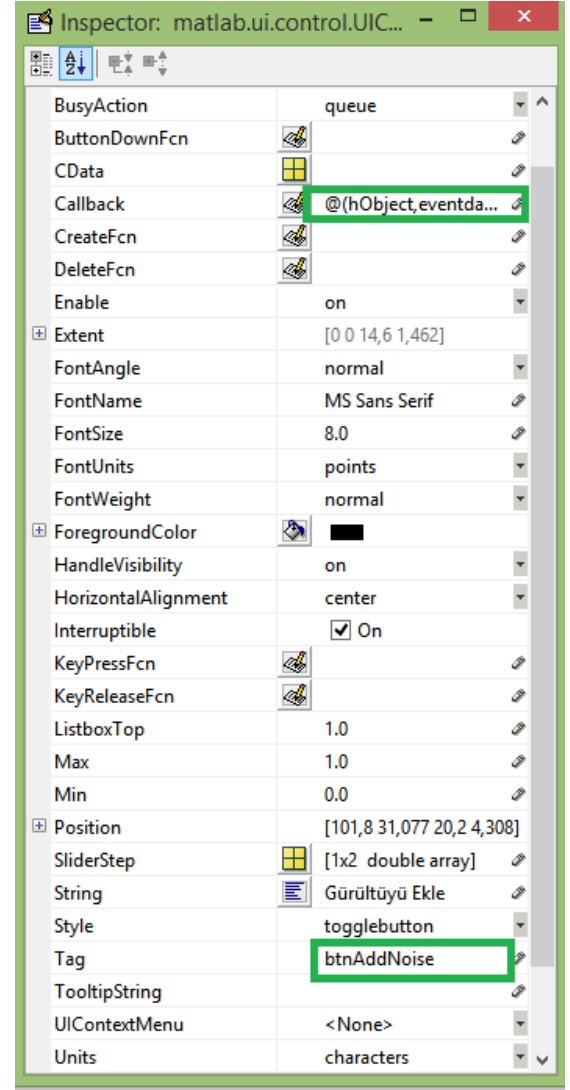
Herhangi bir eleman ile işlem yaparak arka planda kod yürütürüz bu kodlara erişim elemanların property penceresinde callback değişkeninde çalıştıracağı fonksiyonun adresi yer alır.

Örneğin bir butona basıldığında mesaj yazdıran kodu programı yazalım

Şekil1 deki gibi butonumuzu çizdirdikten sonra butonun üzerine çift tıklayalım ve property penceresinin açılacağını göreceğiz.

Sağdaki şekilde property penceresi görülmektedir. Bu pencerenin içinde callback özelliğinin içi otomatik olarak yazılmış çünkü içeriği tag adına göre otomatik yazılır. Biz callback özelliğini değiştirmeyiz sadece tag adını değiştirip formu kaydetmemiz sonucu fonksiyon tag adıyla kaydolur.

Callback özelliğinin solunda bulunan  iconunu tıkladığımızda bizi callback fonksiyonuna atar ve içeriğini yazarız.



```
% --- Executes on button press in btnAddNoise.
function btnAddNoise_Callback(hObject, eventdata, handles)
if(get(handles.btnAddNoise, 'value'))
handles.inimageNoise = imnoise(handles.inimage, 'salt & pepper');
handles.inimage = handles.inimageNoise ;
axes(handles.axes1);
imshow(handles.inimage);
set(handles.btnAddNoise, 'string', 'Gürültüyü Çıkart');
else
handles.inimage =handles.orjimage ;
```

Matlabta Kullanılan Filtreleme Fonksiyonları

Yumuşatma

Resimde yumuşatma için average (ortalama filtre) veya gaussian filtresi(Ağırlıklı Ortalama Filtresi) Kullanılabilir.

Average Filtresi-(imfilter Fonksiyonu)

imfilter fonksiyonu korelasyon çekirdeği ile resmin tamamına dolaşarak resmi filtreleyen fonksiyondur.İmfilter sayesinde istediğimiz giriş çekirdeğiyle resmimize filtre uygulayabiliriz.Average filtresini imfilter ve birler matrisi ile sağlayabiliriz.

```
h = ones(5,5) / 5^2;
```

burada 5'e 5l ik birim matris oluşturulup 25 e bölünmüştür bu %sayede kolerasyon işlemi sonucu matrisin ortalaması alınacaktır

```
handles.outimage=imfilter(handles.inimage,h);
```

yukardaki kodda ise oluşturduğumuz h birim matrsinin giriş resimler birlikte imfilter fonksiyonu kullanıldığı görülmektedir.bu sayede kolerasyon çekirdeği h giriş resmi inimage işlemlere tabi tutulup çıkış resmi outimage şeklinde sonuca varıcaaktır.

```
imshow(handles.outimage);
```

ve outimage gösterilecektir ve aşağıdaki bir giriş resim çıkış resmi gibi görülecektir.



İmgaussfilt Fonksiyonu

Matlabın İmgaussfilt fonksiyonu görüntüye gaussian filtresi (ağırlıklı ortalama filtresi) uygulayan fonksiyondur. Giriş resmi ve standart sapma değışkeni ile fonksiyonun parametreleridir ve bize çıkış resmini üretir.

```
handles.outimage = imgaussfilt(handles.inimage, 2);
```

```
imshow(handles.outimage);
```

Yukarıdaki Kodda 1.satırda giriş resmi imgaussfilt fonksiyonuna parametre olarak 2 standart sapmasıyla birlikte işleme tabi tutulup outimage değişkeni olarak çıkışı sağlanmıştır.ve imshow ile gösterilmiştir.



Yukarıdaki resmine standart sapması 2 olan imgaussfilt (yukarıdaki kod) uygulanmıştır ve sonucu çıkış resminde görülmektedir.

Keskinleştirme

Kolerasyon ile birçok filtreleme yöntemi sağlandığı gibi biri de keskinleştirme yöntemi unsharp filtresi yöntemidir.unsharp filtresi kolerasyon çekirdeğinin bir şeklidir.

UnSharp Filtresi

Matlabın imfilter fonksiyonu ile unsharp filtresi uygulayabiliyoruz

```
h = fspecial('unsharp');
```

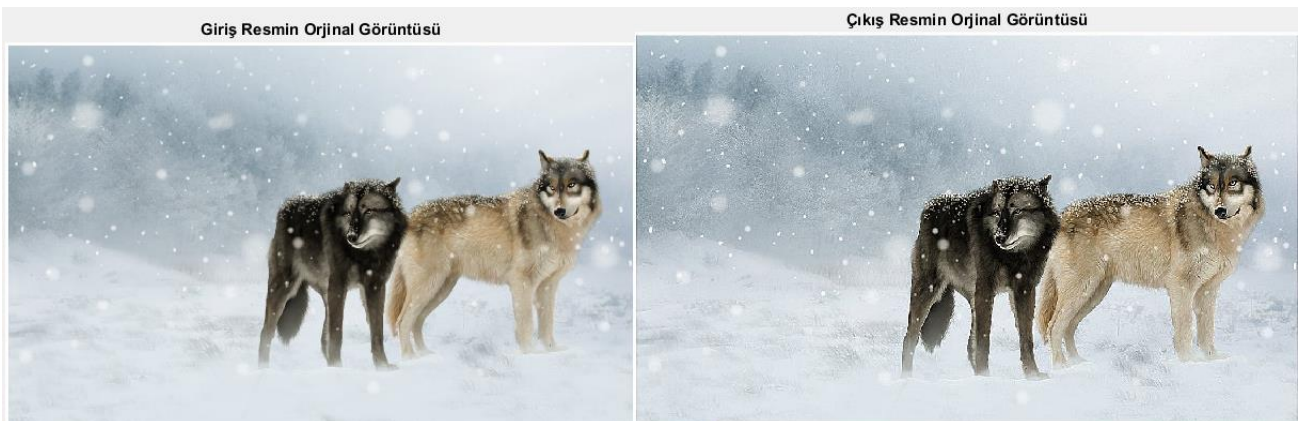
fspecial('unsharp') unsharp matrisini oluşturan fonksiyon.

```
handles.outimage=imfilter(handles.inimage,h);
```

oluşturduğumuz unsharp matrisini imfilter ile çekirdek olarak giriş resmimize uyguluyoruz ve çıkış resmimizi elde ediyoruz.

```
imshow(handles.outimage);
```

sonuç olarak çıkış resmini imshow ile gösteriyoruz.Resimde de görüldüğü gibi çıkış elde ediyoruz.



Kenar Bulma

Log Filtresi

Bu filtreye Marr-Hildreth (Laplacian of Gaussian – LoG) algoritması denir. Gaussian filtresine Laplasi alınarak işlem yapar.

```
h = fspecial('log');
```

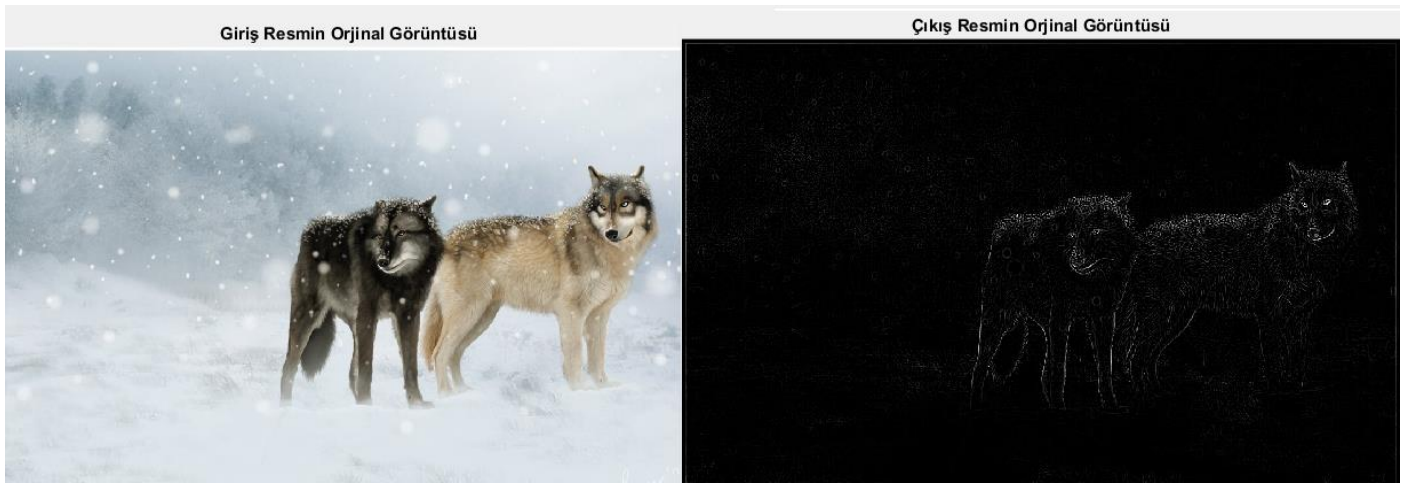
fspecial('log') fonksiyonu log çekirdek matrisini oluşturdu ve h değişkenine atadı.

```
handles.outimage=imfilter(handles.inimage,h);
```

ve imfilter ile log filtremizi resimden geçirerek çıkış resmimizi elde etmiş bulunmaktayız.

```
imshow(handles.outimage);
```

son olarak çıkış resmimizi gösteriyoruz ve resimdeki gibi çıkı görüntüsü elde ediyoruz.



Canny Filtresi

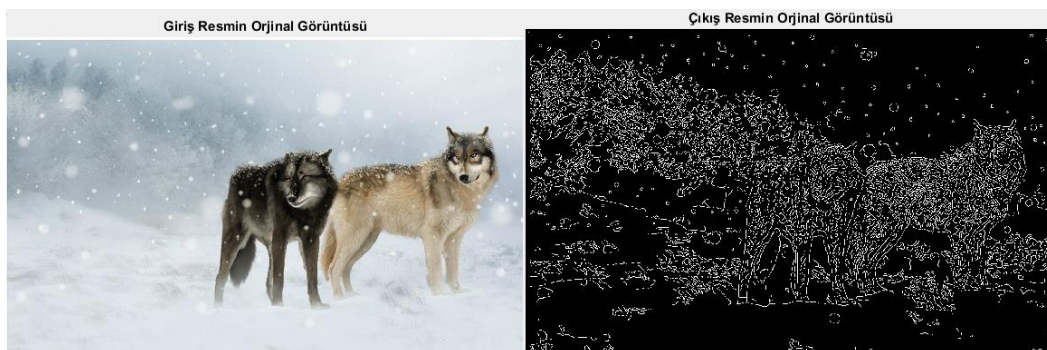
Kenar bulmada son derece etkin bir algoritmadır. Önce görüntüdeki gürültü bir sigma değerine göre üretilen Gaussian çekirdek ile konvolusyonu alınarak azaltılır. Daha sonra, gradyent operatörü uygulanarak, kenar gradyent büyüklüğü ve yönü hesaplanır. Kenarlar, non maxima baskılama uygulanarak inceltilir. Son olarak görüntü, ikili eşikleme uygulanarak istenmeyen ayrıntılardan arındırılır.

```
handles.outimage = edge(handles.ingrayImage,'canny');
```

matlab edge fonksiyonu kenar bulma fonksiyonu fakat grisinde gray scala imge girilmesi gereklidir. Çıkışında bitmap image üretir. Buradaki fonksiyonda ikinci parametre olarak canny stringi girilerek canny çekirdek matrisi ile giriş resmi işleme tabi tutulup sonucu outimage değişkenine atılmıştır.

```
imshow(handles.outimage);
```

ve sonuç gösterilmiştir.



Roberts Filtresi

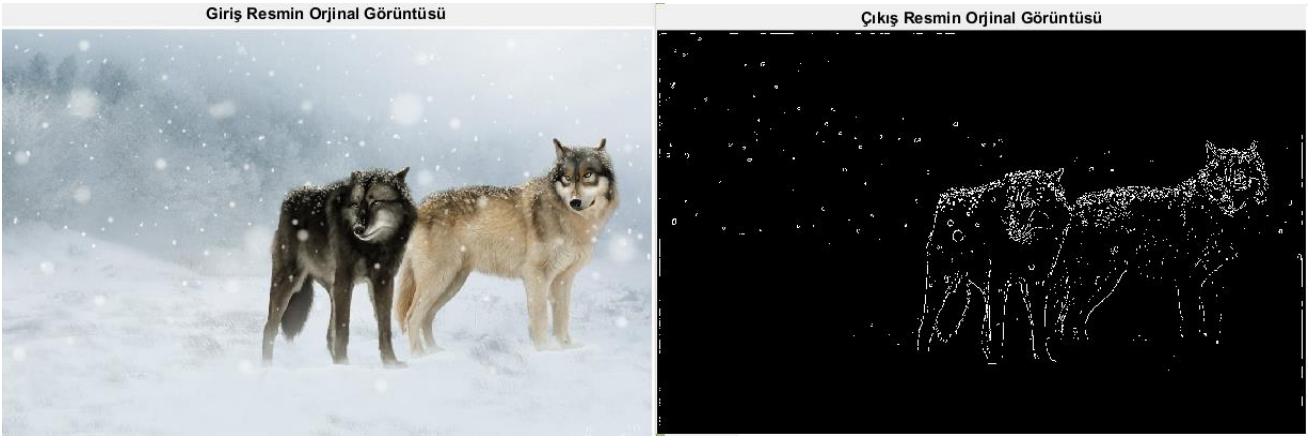
Bu filtre diagonal olarak kenar tarar. Kernel matrisi şöyledir: $\begin{bmatrix} 1,1,0;1,0,-1;0,-1,-1 \end{bmatrix}$ veya $\begin{bmatrix} 2,1,0;1,0,-1;0,-1,-2 \end{bmatrix}$.

```
handles.outimage = edge(handles.ingrayImage,'Roberts');
```

edge fonksiyonu canny de kullanıldığı gibi roberts filtresi içinde kullanılmıştır ve çıkış resmi outimage olarak atanmıştır.

```
imshow(handles.outimage);
```

ve görüntü gösterilmiştir.



Prewitt Filtresi

Bu filtrede sobel filtresi gibi düşey ve yatay keskinlik yakalar. Kernel matrisi sobelden farklıdır. Matrisin dizimi şöyledir: $\begin{bmatrix} 1,1,1;0,0,0;-1,-1,-1 \end{bmatrix}$ veya $\begin{bmatrix} 1,0,-1;1,0,-1;1,0,-1 \end{bmatrix}$.

Prewitt filtresini renkli giriş resmi için imfilter fonksiyonu olarak yapabiliriz. veya edge fonksiyonu ile gray skalasını giriş olarak sonucu elde edebiliriz.iki yöntemde aynı sonucu elde edecektir.Biz imfilter fonksiyonunu kullanalım.r g b kanalları için ayrı görüntüler elde edelim

```
h = [1,1,1; 0,0,0 ; -1,-1,-1];
```

h çekirdek matrisi oluşturuluyor.

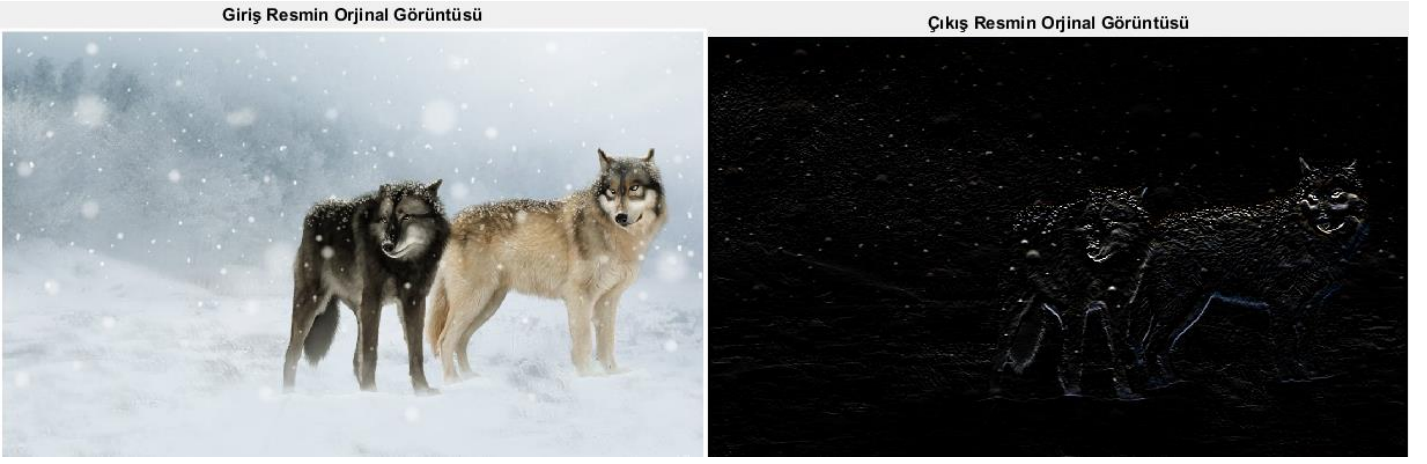
```
handles.outimage=imfilter(handles.inimage,h);
```

imfilter ile oluşturulan çekirdek matrisi ve giriş resmi işleme tabi tutulup sonucu outimage e atanıyor.

```
imshow(handles.outimage)
```

ve sonuç gösteriliyor.

Aşağıda prewitt filtresi uygulanan giriş resmin çıkış resmi görülmektedir.



Sobel Filtresi

Sobel operatörü yatay ve düşey yönde keskinlikleri yakalar. Kernel matrisi dizilimi şöyledir: $[1,2,1;0,0,0;-1,-1,-1]$ veya $[1,0,-1;2,0,-2;-1,-2,-1]$.

```
handles.outimage = edge(handles.ingrayImage,'Sobel');
```

edge fonksiyonunun 2. Parametresi Sobel String ifadesiyle girişi sonucu gray scala olarak giriş resmine sobel filtresi işlemini uygular ve çıkışını outimage değişkenine atar.

```
imshow(handles.outimage);
```

ve resmin gösterilmesi.



Sonuc olarak soldaki girişe sağdaki çıkış elde edilmiştir.

Laplace Filtresi

Laplace operatörü her yöndeki keskinleştirme yapmaya yarar.

```
h = fspecial('laplacian');
```

fspecial('laplacian') fonksiyonu laplace matrisi üretmiş ve h değişkenine atamıştır.

```
handles.outimage=imfilter(handles.inimage,h);
```

```
imshow(handles.outimage);
```

diğer imfilter fonksiyonları gibi aynı işlemler gerçekleşmektedir sadece giriş çekirdek matrisi farklıdır.

Giriş Resmin Orjinal Görüntüsü



Çıkış Resmin Orjinal Görüntüsü



Yukarıda Giriş Resme laplace filtresi uygulanıp çıkış resmi oluşturulmuştur.

Kabartma

Resimde kabarık bir görünüm elde etmek için konvolüsyon çekirdeğini $[0, -3, 0; 0, 1, 0; 0, 3, 0]$ şeklinde ayarlarız.ve imfilter ile resmin tamamına uygularız.

$$h = [0, -3, 0; 0, 1, 0; 0, 3, 0] ;$$

kabartacağımız görüntü için çekirdek matrisimiz h olarak ayarlıyoruz.

```
handles.outimage=imfilter(handles.inimage,h);
```

Burada imfilter fonksiyonuna giriş resmimiz ve çekirdek matrisimiz giriyor.

```
imshow(handles.outimage);
```

son olarak görüntü alıyoruz.

Giriş Resmin Orjinal Görüntüsü



Çıkış Resmin Orjinal Görüntüsü



Yukardaki Görüntü giriş olarak kabartma filtresi uygulanmış resim çıkış olarak görülmektedir.

Gürültü Giderme

Gürültü Ekleme (imnoise fonksiyonu)

Matlabta gürültüsüz olan bir resime gürültü eklemek için imnoise fonksiyonu kullanılır.

```
imnoise(handles.inimage,'salt & pepper',0.02)
```

Giriş Resime Tuz Biber Gürültüsü Eklenmiştir.Tuz Biber Gürültüsünden farklı birkaç çeşit gürültüde eklenebilmektedir.

İmnoise ile daha fazla bilgi için <https://www.mathworks.com/help/images/ref/imnoise.html> adresini inceleyebilirsiniz.

Medyan Filtresi(medFilt)

Gürültü gidermede kullanılan yöntemlerden biri giriş resime medyan filtresi uygulanmasıdır. Medyan filtresi uygulanan görüntüdeki tuz biber gürültüsüne benzeyen küçük noktalar kaybolur.

```
handles.outimage = medfilt2(handles.outgrayImage);
```

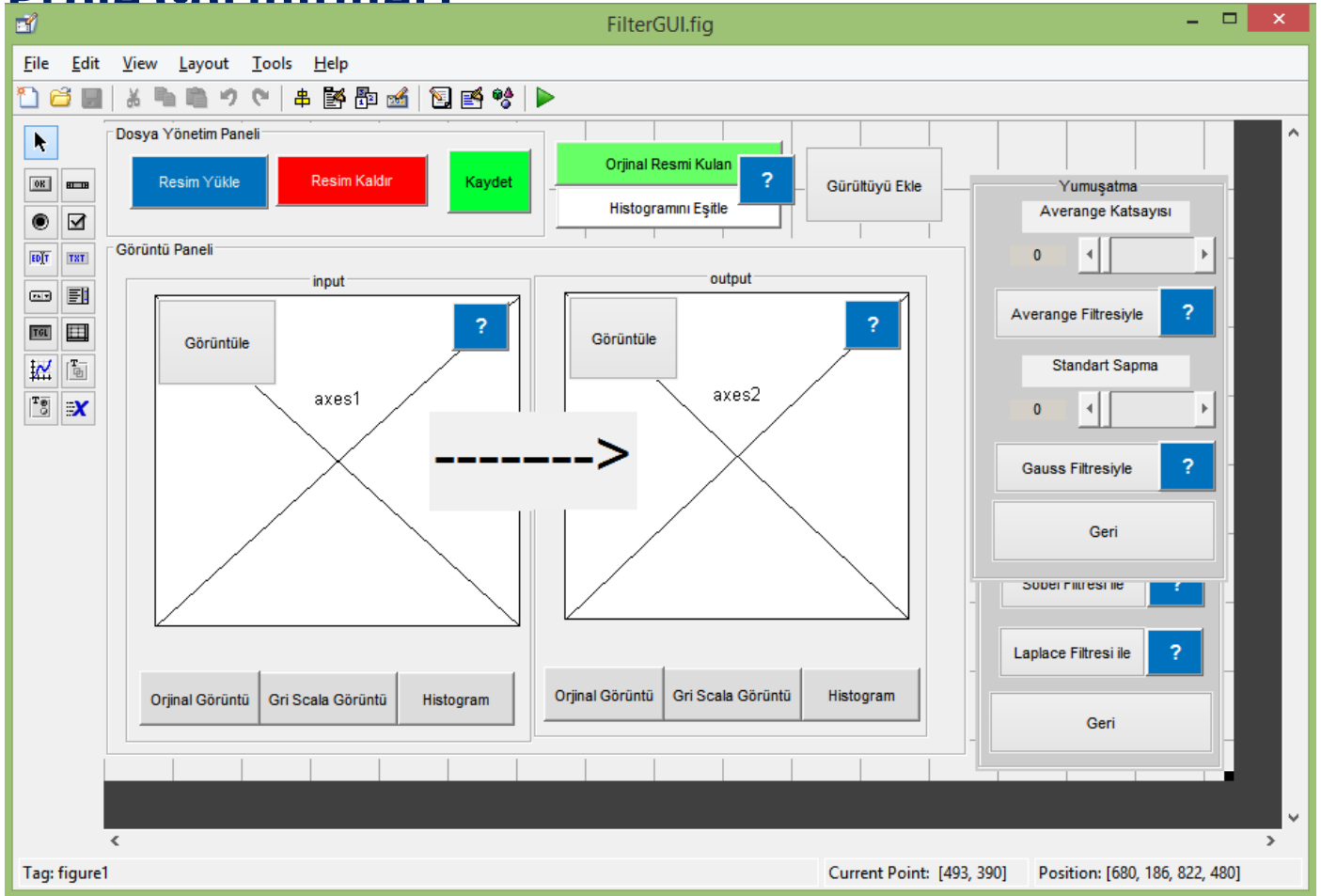
Gürültü gidermek için medFilt2 Fonksiyonunu kullanarak giriş resimdeki tuz biber gürültüsünü giderilmiş olarak çıkış resimde elde etmiş olduk.

```
imshow(handles.outimage);
```

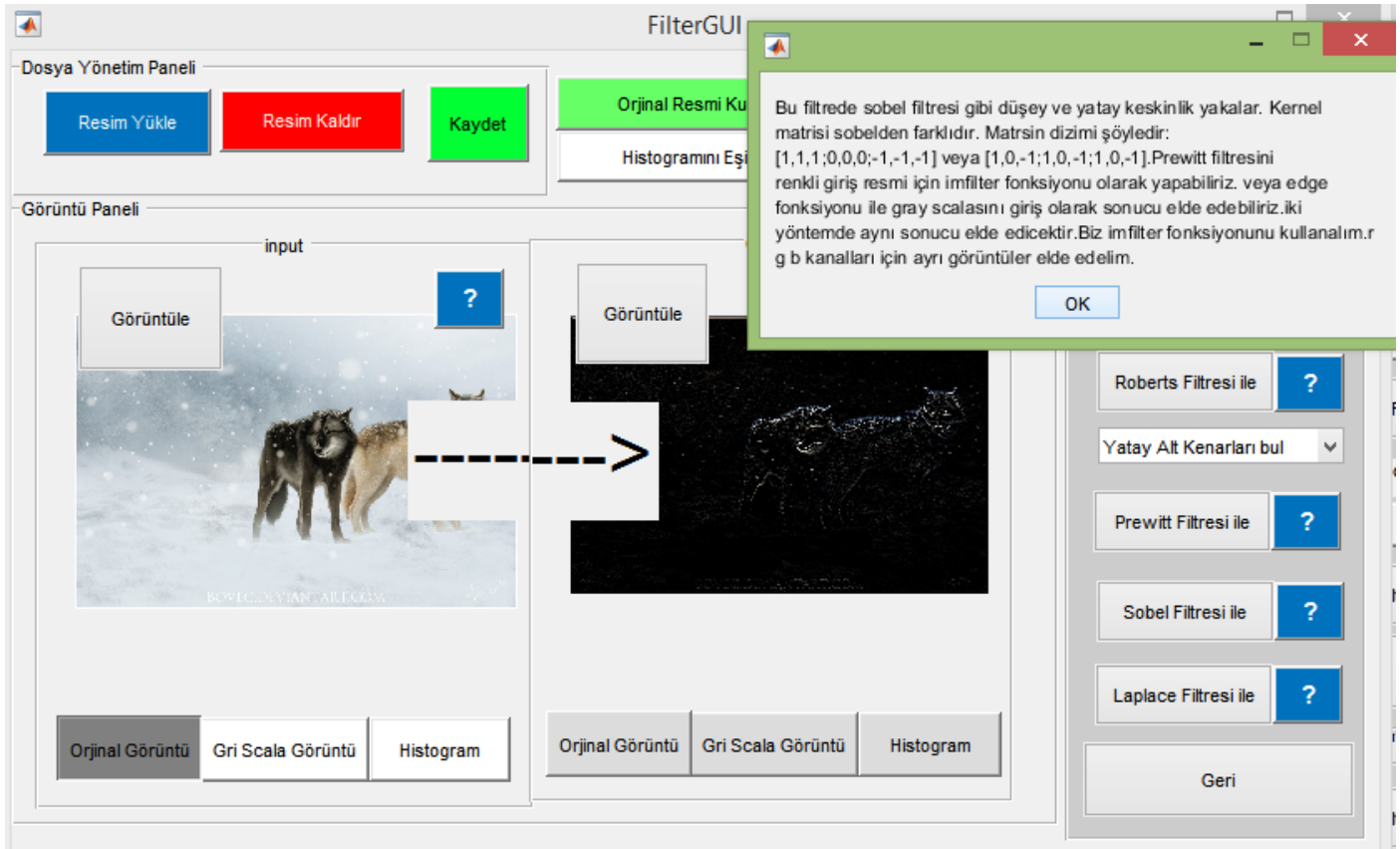
ve resmimizi imshow ile gösterdik.sonuç olarak aşağıdaki giriş resmine medyan filtresi uygulayarak çıkış resmini elde etmiş olduk.



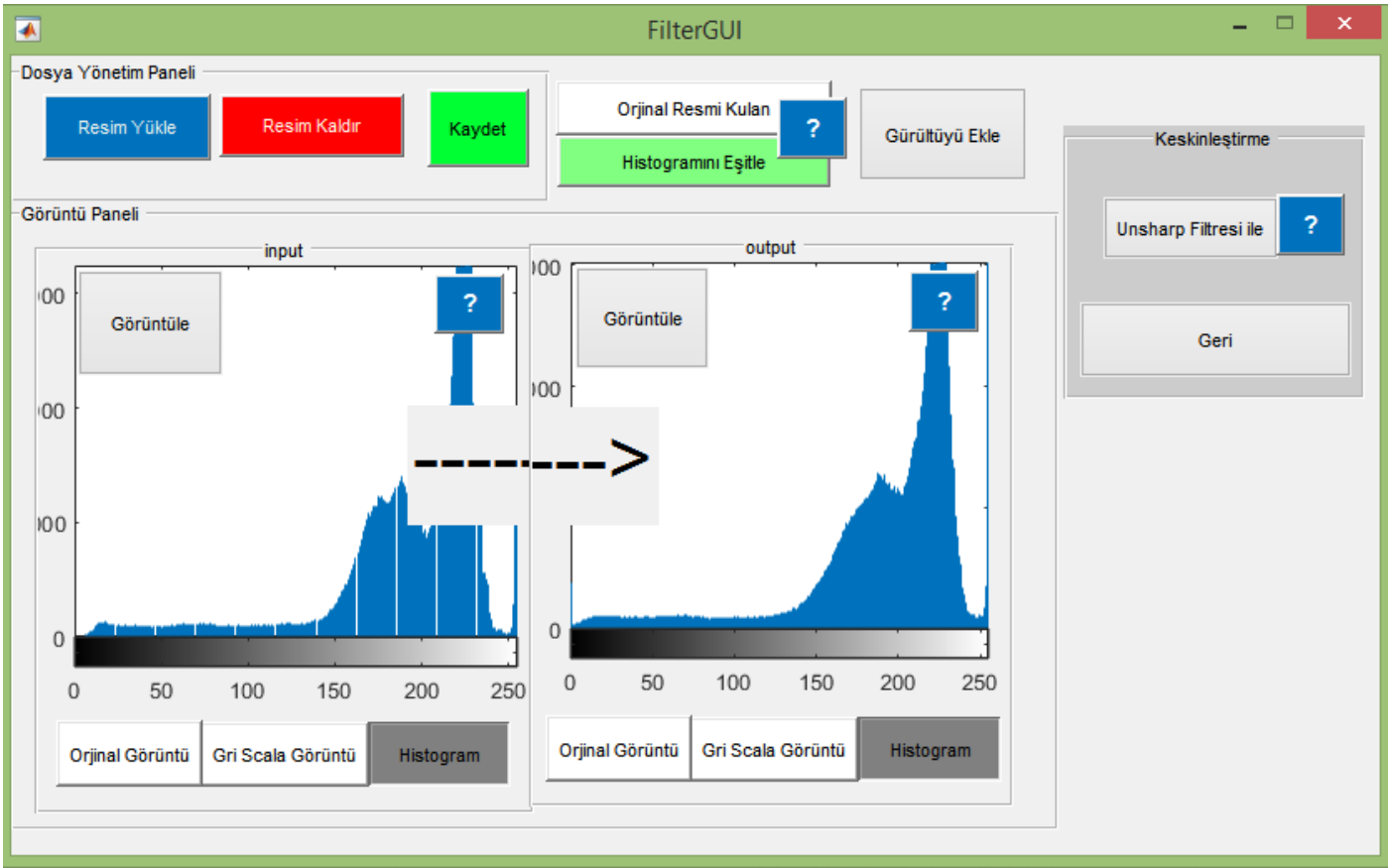
Proje Görüntüleri



FilterGUI uygulamasının ekran görüntüsü



Uygulamanın soru işaretine tıklanması sonucu bilgi vermesi



Uygulama resimler arasındaki farkları görebilmek için işlemin histogram görünümü

Kodlar

```
function varargout = FilterGUI(varargin)
% FILTERGUI MATLAB code for FilterGUI.fig
%   FILTERGUI, by itself, creates a new FILTERGUI or raises the existing
%   singleton*.
%
%   H = FILTERGUI returns the handle to a new FILTERGUI or the handle to
%   the existing singleton*.
%
%   FILTERGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in FILTERGUI.M with the given input arguments.
%
%   FILTERGUI('Property','Value',...) creates a new FILTERGUI or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before FilterGUI_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to FilterGUI_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to txthelp FilterGUI

% Last Modified by GUIDE v2.5 18-Dec-2016 18:15:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @FilterGUI_OpeningFcn, ...
```

```

        'gui_OutputFcn', @FilterGUI_OutputFcn, ...
        'gui_LayoutFcn', [] , ...
        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before FilterGUI is made visible.
function FilterGUI_OpeningFcn(hObject, eventdata, handles, varargin)
    axes(handles.axes1);
    imshow('');

    axes(handles.axes2);
    imshow('');
    set(handles.pnlIn , 'visible' , 'off');
    set(handles.panelOut , 'visible' , 'off');
    set(handles.toggleOriginalImage1, 'enable', 'inactive');
    set(handles.toggleOriginalImage1, 'value', 1);

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes FilterGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = FilterGUI_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

% --- Resim Yükleme butonun fonksiyonu
function loadImage_Callback(hObject, eventdata, handles)

%File Dialog penceresi açılıp resim aldığımız bölüm
[filename, pathname] = ...
    uigetfile({'*.png'; '*.jpg';}, 'Filtrenecek Resmi Seçiniz', 'C:\Pictures');
IMAGE = imread(fullfile(pathname, filename));
axes(handles.axes1);
imshow(IMAGE);
axes(handles.axes2);
imshow(IMAGE);
handles.ingrayImage = rgb2gray(IMAGE);
handles.outgrayImage = rgb2gray(IMAGE);
handles.orjimage = IMAGE ;
handles.inimage = IMAGE ;
handles.outimage = IMAGE ;
set(handles.pnlIn , 'visible' , 'on');
set(handles.panelOut , 'visible' , 'on');
guidata(hObject, handles)

% --- Executes on button press in pushbutton2.

```

```

function pushbutton2_Callback(hObject, eventdata, handles)
%Resim kaldırma buton işlemleri
    IMAGE = '';
    handles.inimage = '' ;
    handles.ingrayImage = '' ;
    handles.outimage = '';
    axes(handles.axes1);
    imshow(IMAGE);
    axes(handles.axes2);
    imshow(IMAGE);
    %Componentlerin içerileri set fonksiyonu ile değiştiriliyor.
    set(handles.pnlIn , 'visible' , 'off');
    set(handles.panelOut , 'visible' , 'off');
    %eğer handles nesnesine değişken eklediysek guidata fonk ile kaydetmemiz gerekir.
    guidata(hObject,handles);

```

```

% --- Executes on button press in toggleOriginalImage1.
function toggleOriginalImage1_Callback(hObject, eventdata, handles)
    if(get(handles.toggleOriginalImage1,'value') == 1)
        set(handles.toggleGrayImage1,'enable','on');
        set(handles.toggleInhistogram1,'enable','on');

        set(handles.toggleGrayImage1,'background',[1 1 1]);
        set(handles.toggleInhistogram1,'background',[1 1 1]);
        set(handles.toggleOriginalImage1,'background',[0.5 0.5 0.5]);

        axes(handles.axes1);
        imshow(handles.inimage);
        set(handles.toggleGrayImage1,'value',0);
        set(handles.toggleInhistogram1,'value',0);
        guidata(hObject,handles)
        set(handles.toggleOriginalImage1,'enable','inactive');

    end

```

```

% --- Executes on button press in toggleGrayImage1.
function toggleGrayImage1_Callback(hObject, ~, handles)
    if(get(handles.toggleGrayImage1,'value') == 1)
        set(handles.toggleOriginalImage1,'enable','on');
        set(handles.toggleInhistogram1,'enable','on');

        set(handles.toggleGrayImage1,'background',[0.5 0.5 0.5]);
        set(handles.toggleOriginalImage1,'background',[1 1 1]);
        set(handles.toggleInhistogram1,'background',[1 1 1]);

        axes(handles.axes1);
        handles.ingrayImage = rgb2gray(handles.inimage);
        imshow(handles.ingrayImage);
        set(handles.toggleOriginalImage1,'value',0);
        set(handles.toggleInhistogram1,'value',0);
        guidata(hObject,handles)
        set(handles.toggleGrayImage1,'enable','inactive');

    end

```

```

% --- Executes on button press in toggleInhistogram1.
function toggleInhistogram1_Callback(hObject, ~, handles)
    if(get(handles.toggleInhistogram1,'value') == 1)
        set(handles.toggleOriginalImage1,'enable','inactive');
        set(handles.toggleOriginalImage1,'enable','on');
        set(handles.toggleGrayImage1,'enable','on');

        set(handles.toggleOriginalImage1,'background',[1 1 1]);

```



```

set(handles.toggleGrayImage1,'background',[1 1 1]);
set(handles.toggleInhistogram1,'background',[0.5 0.5 0.5]);

axes(handles.axes1);
imshow(handles.ingrayImage);
set(handles.toggleOriginalImage1,'value',0);
set(handles.toggleGrayImage1,'value',0);
guidata(hObject,handles)
set(handles.toggleInhistogram1,'enable','inactive');

axes(handles.axes1);
imhist(handles.ingrayImage);
end

% --- Executes on button press in btnShowAxes1.
function btnShowAxes1_Callback(hObject, eventdata, handles)
figure('name','Görüntüle');
if(get(handles.toggleInhistogram1,'value'))
    imhist(handles.ingrayImage), title('Giriş Resmin Histogram grafiği'),
elseif(get(handles.toggleGrayImage1,'value'))
    imshow(handles.ingrayImage),title ('Giriş Resmin Gray Scalası');
else
    imshow(handles.inimage) , title ('Giriş Resmin Orjinal Görüntüsü');
end

% --- Executes on button press in btnShowAxes2.
function btnShowAxes2_Callback(hObject, eventdata, handles)
figure('name','Görüntüle');
if(get(handles.toggleInhistogram2,'value'))
    imhist(handles.outgrayImage), title('Çıkış Resmin Histogram grafiği'),
elseif(get(handles.toggleGrayImage2,'value'))
    imshow(handles.outgrayImage),title ('Çıkış Resmin Gray Scalası');
else
    imshow(handles.outimage) , title ('Çıkış Resmin Orjinal Görüntüsü');
end

% --- Executes on button press in btnAdjustImage.
function btnAdjustImage_Callback(hObject, eventdata, handles)
handles.adjimage = imadjust(handles.ingrayImage);
handles.ingrayImage = handles.adjimage ;
set(handles.btnAdjustImage,'background',[0.5 1 0.5]);
set(handles.btnUseOriginal,'background',[1 1 1]);
guidata(hObject,handles);

if(get(handles.toggleInhistogram1,'value') == 1)
    axes(handles.axes1);
    imhist(handles.adjimage);
elseif(get(handles.toggleGrayImage1,'value') == 1)
    axes(handles.axes1);
    imshow(handles.adjimage);
else
    if(get(handles.toggleGrayImage1,'value') == 1)
        set(handles.toggleOriginalImage1,'enable','on');
        set(handles.toggleInhistogram1,'enable','on');

        set(handles.toggleGrayImage1,'background',[0.5 0.5 0.5]);
        set(handles.toggleOriginalImage1,'background',[1 1 1]);
        set(handles.toggleInhistogram1,'background',[1 1 1]);
    end
end
end

```

```

% --- Executes on button press in btnUseOriginal.
function btnUseOriginal_Callback(hObject, eventdata, handles)
handles.ingrayImage = rgb2gray(handles.orjimage);
set(handles.btnAdjustImage, 'background', [1 1 1]);
set(handles.btnUseOriginal, 'background', [0.5 1 0.5]);
guidata(hObject, handles);

if(get(handles.toggleInhistogram1, 'value') == 1)
    axes(handles.axes1);
    imhist(handles.ingrayImage);
elseif(get(handles.toggleGrayImage1, 'value') == 1)
    axes(handles.axes1);
    imshow(handles.ingrayImage);
else
    if(get(handles.toggleGrayImage1, 'value') == 1)
        set(handles.toggleOriginalImage1, 'enable', 'on');
        set(handles.toggleInhistogram1, 'enable', 'on');

        set(handles.toggleGrayImage1, 'background', [0.5 0.5 0.5]);
        set(handles.toggleOriginalImage1, 'background', [1 1 1]);
        set(handles.toggleInhistogram1, 'background', [1 1 1]);
    end
end

% --- Executes on button press in btnSmooth.
function btnSmooth_Callback(hObject, eventdata, handles)
    set(handles.pnlSmooth, 'visible', 'on');
    set(handles.pnlFiltre, 'visible', 'off');

% --- Executes on button press in btnKeskin.
function btnKeskin_Callback(hObject, eventdata, handles)
    set(handles.pnlKeskin, 'visible', 'on');
    set(handles.pnlFiltre, 'visible', 'off');

function btnKenarBul_Callback(hObject, eventdata, handles)
    set(handles.pnlKenarBulma, 'visible', 'on');
    set(handles.pnlFiltre, 'visible', 'off');

% --- Executes on button press in btnKabartConv.
function btnKabart_Callback(hObject, eventdata, handles)
    set(handles.pnlKabart, 'visible', 'on');
    set(handles.pnlFiltre, 'visible', 'off');

% --- Executes on button press in btnHelp3.
function btnHelp3_Callback(hObject, eventdata, handles)
msgbox('Original Resim yada hisyogramını eşitle işlem yap');

% --- Executes on button press in btnSmthConv.
function btnSmthConv_Callback(hObject, eventdata, handles)
value =get(handles.sliderSmooth, 'value');
h = ones(value,value) / value^2;
handles.outimage=imfilter(handles.inimage,h);
axes(handles.axes2);
imshow(handles.outimage);
guidata(hObject, handles);

% --- Executes on button press in btnHelp4.
function btnHelp4_Callback(hObject, eventdata, handles)
% hObject      handle to btnHelp4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

```

```

% --- Executes on button press in btnHelp5.
function btnHelp5_Callback(hObject, eventdata, handles)
    msgbox('imfilter fonksiyonu korelasyon çekirdeği ile resmin tamamına dolaşarak resmi filtreleyen fonksiyondur.İmfilter sayesinde istediğimiz giriş çekirdeğiyle resmimize filtre uygulayabiliriz.Average filtresini imfilter ve birler matrisi ile sağlayabiliriz.');
```

```

function btnPrewittFilter_Callback(hObject, eventdata, handles)
yatay = get(handles.pmFindEdge , 'value');
%set(handles.txthelp,'string',yatay);
if(yatay == 1)
    h = [1,0,-1; 1,0,-1 ; 1,0,-1];
elseif(yatay == 2)
    h = [-1,0,1; -1,0,1 ; -1,0,1];
elseif(yatay == 3)
    h = [-1,-1,-1; 0,0,0 ; 1,1,1];
elseif(yatay == 4)
    h = [1,1,1; 0,0,0 ; -1,-1,-1];
end
handles.outimage=imfilter(handles.inimage,h);
axes(handles.axes2);
imshow(handles.outimage);
guidata(hObject,handles);

% --- Executes on button press in btnHelp7.
function btnHelp7_Callback(hObject, eventdata, handles)
msgbox('Bu filtrede sobel filtresi gibi düşey ve yatay keskinlik yakalar. Kernel matrisi sobelden farklıdır. Matrsin dizimi şöyledir: [1,1,1;0,0,0;-1,-1,-1] veya [1,0,-1;1,0,-1;1,0,-1].Prewitt filtresini renkli giriş resmi için imfilter fonksiyonu olarak yapabiliriz. veya edge fonksiyonu ile gray scalasını giriş olarak sonucu elde edebiliriz.iki yöntemde aynı sonucu elde edecektir.Biz imfilter fonksiyonunu kullanalım.r g b kanalları için ayrı görüntüler elde edelim.');
```

```

function btnUnSharp_Callback(hObject, eventdata, handles)
h = fspecial('unsharp');
handles.outimage=imfilter(handles.inimage,h);
axes(handles.axes2);
imshow(handles.outimage);
guidata(hObject,handles);

% --- Executes on button press in btnHelp6.
function btnHelp6_Callback(hObject, eventdata, handles)
msgbox('Kolerasyon ile birçok filtreleme yöntemi sağlandığı gibi biri de keskinleştirme yöntemi unsharp filtresi yöntemidir.unsharp filtresi kolerasyon çekirdeğinin bir şeklidir.');
```

```

% --- Executes on button press in btnHelp2.
function btnHelp2_Callback(hObject, eventdata, handles)
msgbox('Çıkış Resmin Görüntüsü');
```

```

function btnHelp1_Callback(hObject, eventdata, handles)
msgbox('Giriş Resmin Görüntüsü');
```

```

% --- Executes on button press in btnGeril.
function btnGeril_Callback(hObject, eventdata, handles)
    set(handles.pnlSmooth,'visible','off');
    set(handles.pnlFiltre,'visible','on');
```

```

% --- Executes on button press in btnKeskin.
function pushbutton29_Callback(hObject, eventdata, handles)
% hObject      handle to btnKeskin (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
```

```

% handles      structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton30.
function pushbutton30_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton30 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% --- Executes on button press in btnGeri2.
function btnGeri2_Callback(hObject, eventdata, handles)
    set(handles.pnlKeskin, 'visible', 'off');
    set(handles.pnlFiltre, 'visible', 'on');

function btnGeri3_Callback(hObject, eventdata, handles)
    set(handles.pnlKenarBulma, 'visible', 'off');
    set(handles.pnlFiltre, 'visible', 'on');

% --- Executes on button press in btnKabartConv.
function pushbutton34_Callback(hObject, eventdata, handles)
% hObject      handle to btnKabartConv (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton35.
function pushbutton35_Callback(hObject, eventdata, handles)
msgbox('Resimde kabarık bir görünüm elde etmek için konvolüsyon çekirdeğini [0,-3,0;0,1,0;0,3,0] şeklinde ayarlarız.ve imfilter ile resmin tamamına uygularız.');
```

```

function btnGeri4_Callback(hObject, eventdata, handles)
    set(handles.pnlKabart, 'visible', 'off');
    set(handles.pnlFiltre, 'visible', 'on');

% --- Executes on button press in toggleInhistogram2.
function toggleInhistogram2_Callback(hObject, eventdata, handles)
    if (get(handles.toggleInhistogram2, 'value') == 1)
        set(handles.toggleOriginalImage2, 'enable', 'inactive');
        set(handles.toggleOriginalImage2, 'enable', 'on');
        set(handles.toggleGrayImage2, 'enable', 'on');

        set(handles.toggleOriginalImage2, 'background', [1 1 1]);
        set(handles.toggleGrayImage2, 'background', [1 1 1]);
        set(handles.toggleInhistogram2, 'background', [0.5 0.5 0.5]);

        axes(handles.axes2);
        imshow(handles.outgrayImage);
        set(handles.toggleOriginalImage2, 'value', 0);
        set(handles.toggleGrayImage2, 'value', 0);
        guidata(hObject, handles)
        set(handles.toggleInhistogram2, 'enable', 'inactive');

        axes(handles.axes2);
        imhist(handles.outgrayImage);
    end

function toggleGrayImage2_Callback(hObject, eventdata, handles)
    if (get(handles.toggleGrayImage2, 'value') == 1)
        set(handles.toggleOriginalImage2, 'enable', 'on');
        set(handles.toggleInhistogram2, 'enable', 'on');
```

```

set(handles.toggleGrayImage2, 'background', [0.5 0.5 0.5]);
set(handles.toggleOriginalImage2, 'background', [1 1 1]);
set(handles.toggleInhistogram2, 'background', [1 1 1]);

handles.outgrayImage = rgb2gray(handles.outimage);
axes(handles.axes2);
imshow(handles.outgrayImage);
set(handles.toggleOriginalImage2, 'value', 0);
set(handles.toggleInhistogram2, 'value', 0);
set(handles.toggleGrayImage2, 'enable', 'inactive');
guidata(hObject, handles);
end

function toggleOriginalImage2_Callback(hObject, eventdata, handles)
if(get(handles.toggleOriginalImage2, 'value') == 1)
    set(handles.toggleGrayImage2, 'enable', 'on');
    set(handles.toggleInhistogram2, 'enable', 'on');

    set(handles.toggleGrayImage2, 'background', [1 1 1]);
    set(handles.toggleInhistogram2, 'background', [1 1 1]);
    set(handles.toggleOriginalImage2, 'background', [0.5 0.5 0.5]);

    axes(handles.axes2);
    imshow(handles.outimage);
    set(handles.toggleGrayImage2, 'value', 0);
    set(handles.toggleInhistogram2, 'value', 0);
    guidata(hObject, handles);
    set(handles.toggleOriginalImage2, 'enable', 'inactive');

end

% --- Executes on slider movement.
function sliderSmooth_Callback(hObject, eventdata, handles)
set(handles.sliderSmooth, 'value', round(get(handles.sliderSmooth, 'value')));
set(handles.txtSmooth, 'string', get(handles.sliderSmooth, 'value'));

function sliderSmooth_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sliderSmooth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject, 'BackgroundColor'), get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', [.9 .9 .9]);
end

function pmFindEdge_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pmFindEdge (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'), get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

```

```

% --- Executes on button press in btnKabartConv.
function btnKabartConv_Callback(hObject, eventdata, handles)
strange = get(handles.sliderKabart, 'value');
status = get(handles.pmKabart, 'value');
if(status == 1)
    h = [0,-strange,0;0,1,0;0,strange,0] ;
elseif(status == 2)
    h = [0,strange,0;0,1,0;0,-strange,0] ;
elseif(status == 3)
    h = [0,0,0;-strange,1,strange;0,0,0] ;
elseif(status == 4)
    h = [0,0,0;strange,1,-strange;0,0,0] ;
end
handles.outimage=imfilter(handles.inimage,h);
axes(handles.axes2);
imshow(handles.outimage);
guidata(hObject,handles);

function pmKabart_Callback(hObject, eventdata, handles)
% hObject      handle to pmKabart (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns pmKabart contents as cell
array
%           contents{get(hObject,'Value')} returns selected item from pmKabart

% --- Executes during object creation, after setting all properties.
function pmKabart_CreateFcn(hObject, eventdata, handles)
% hObject      handle to pmKabart (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
function sliderKabart_Callback(hObject, eventdata, handles)
set(handles.txtKabart, 'string', get(handles.sliderKabart, 'value'));

function sliderKabart_CreateFcn(hObject, eventdata, handles)
% hObject      handle to sliderKabart (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over sliderKabart.
function sliderKabart_ButtonDownFcn(hObject, eventdata, handles)
% hObject      handle to sliderKabart (see GCBO)

```



```

% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% --- Executes on selection change in pmFindEdge.
function pmFindEdge_Callback(hObject, eventdata, handles)
% hObject      handle to pmFindEdge (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns pmFindEdge contents as cell
array
%           contents{get(hObject,'Value')} returns selected item from pmFindEdge

% --- Executes on button press in btnKaydet.
function btnKaydet_Callback(hObject, eventdata, handles)
imwrite(handles.outimage,'çıkışResmi.png');
msgbox('çıkışResmi.png Çıkış Resmi olarak kaydedildi.')
```

```

% --- Executes on button press in btnGaus.
function btnGaus_Callback(hObject, eventdata, handles)
value = get(handles.sliderGaus,'value');
handles.outimage = imgaussfilt(handles.inimage, value);
axes(handles.axes2);
imshow(handles.outimage);
guidata(hObject,handles);

% --- Executes on button press in btnHelp8.
function btnHelp8_Callback(hObject, eventdata, handles)
msgbox('Matlabın İmgaussfilt fonksiyonu görüntüye gaussian filtresi (ağırlıklı ortalama
filtresi) uygulayan fonksiyondur. Giriş resmi ve standart sapma değişkeni ile
fonksiyonun parametreleridir ve bize çıkış resmini üretir.');
```

```

% --- Executes on slider movement.
function sliderGaus_Callback(hObject, eventdata, handles)
set(handles.sliderGaus,'value',round(get(handles.sliderGaus,'value')));
set(handles.txtgaus,'string',get(handles.sliderGaus,'value'));

% --- Executes during object creation, after setting all properties.
function sliderGaus_CreateFcn(hObject, eventdata, handles)
% hObject      handle to sliderGaus (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on button press in btnAddNoise.
function btnAddNoise_Callback(hObject, eventdata, handles)
if(get(handles.btnAddNoise,'value'))
handles.inimageNoise = imnoise(handles.inimage,'salt & pepper',0.02);
handles.inimage = handles.inimageNoise ;
axes(handles.axes1);
imshow(handles.inimage);
set(handles.btnAddNoise,'string','Gürültüyü Çıkart');
else
    handles.inimage =handles.orjimage ;
    axes(handles.axes1);
    imshow(handles.inimage);
    set(handles.btnAddNoise,'string','Gürültüyü Ekle');
end

```

```

guidata(hObject,handles);

function btnMedian_Callback(hObject, eventdata, handles)
    set(handles.pnlGg, 'visible', 'on');
    set(handles.pnlFiltre, 'visible', 'off');

% --- Executes on button press in btnLogFilter.
function btnLogFilter_Callback(hObject, eventdata, handles)
h = fspecial('log');
handles.outimage=imfilter(handles.inimage,h);
axes(handles.axes2);
imshow(handles.outimage);
guidata(hObject,handles);

% --- Executes on button press in pushbutton44.
function pushbutton44_Callback(hObject, eventdata, handles)
    msgbox('Bu filtreye Marr-Hildreth (Laplacian of Gaussian - LoG) algoritması denir. Gaussian filtresine Laplasi alınarak işlem yapar.');
```

function btnCannyFilter_Callback(hObject, eventdata, handles)

handles.outimage = edge(handles.ingrayImage, 'canny');

axes(handles.axes2);

imshow(handles.outimage);

guidata(hObject,handles);

% --- Executes on button press in pushbutton46.

function pushbutton46_Callback(hObject, eventdata, handles)

msgbox('Kenar bulmada son derece etkin bir algoritmadır. Önce görüntüdeki gürültü bir sigma değerine göre üretilen Gaussian çekirdekle konvolusyonu alınarak azaltılır. Daha sonra, gradyent operatörü uygulanarak, kenar gradyent büyüklüğü ve yönü hesaplanır. Kenarlar, non maxima baskılama uygulanarak inceltirilir. Son olarak görüntü, ikili eşikleme uygulanarak istenmeyen ayrıntılardan arındırılır.');

function btnRobertsFilter_Callback(hObject, eventdata, handles)

handles.outimage = edge(handles.ingrayImage, 'Roberts');

axes(handles.axes2);

imshow(handles.outimage);

guidata(hObject,handles);

function pushbutton48_Callback(hObject, eventdata, handles)

msgbox('Bu filtre diagonal olarak kenar tarar. Kernel matrisi şöyledir: [1,1,0;1,0,-1;0,-1,-1] veya [2,1,0;1,0,-1;0,-1,-2].');

function btnSobelFilter_Callback(hObject, eventdata, handles)

handles.outimage = edge(handles.ingrayImage, 'Sobel');

axes(handles.axes2);

imshow(handles.outimage);

guidata(hObject,handles);

% --- Executes on button press in pushbutton50.

function pushbutton50_Callback(hObject, eventdata, handles)

msgbox('Sobel operatörü yatay ve düşey yönde keskinlikleri yakalar. Kernel matrisi dizilimi şöyledir: [1,2,1;0,0,0;-1,-1,-1] veya [1,0,-1;2,0,-2;-1,-2,-1].');

function btnLaplaceFilter_Callback(hObject, eventdata, handles)

h = fspecial('laplacian');

handles.outimage=imfilter(handles.inimage,h);

axes(handles.axes2);

imshow(handles.outimage);

guidata(hObject,handles);

% --- Executes on button press in pushbutton52.

function pushbutton52_Callback(hObject, eventdata, handles)

msgbox('Laplace operatörü her yöndeki keskinleştirme yapmaya yarar.');

```
function btnMedyanFilter_Callback(hObject, eventdata, handles)
handles.outimage = medfilt2(handles.outgrayImage);
axes(handles.axes2);
imshow(handles.outimage);
guidata(hObject,handles);
```

```
function pushbutton54_Callback(hObject, eventdata, handles)
msgbox('Gürültü gidermede kullanılan yöntemlerden biri giriş resime medyan filtresi uygulanmasıdır. Medyan filtresi uygulanan görüntüdeki tuz biber gürültüsüne benzeyen küçük noktalar kaybolur. Şimdi giriş görüntüsüne GÜRÜLTÜYÜ EKLE butonundan gürültü ekleyip bu filtreyi uygulayın ve sonucu görün :)');
```

```
function geri5_Callback(hObject, eventdata, handles)
set(handles.pnlGg, 'visible', 'off');
set(handles.pnlFiltre, 'visible', 'on');
```

Kaynakça

(Syf:5-6)-Görüntüde filtreleme Türleri - <http://www.bulentsiyah.com/goruntu-filtreleme-uygulamalari-ve-amaclari-matlab/>