MATLAB İLE NESNE TAKİBİ

Özet: Günümüzde görüntü işleme ile ilgili teknolojiler hızla gelişen ve dünya standartlarını önünde sürükleyen sistemlerin gelişmesini sağlamıştır. Askeri, sivil, bilimsel amaçlı birçok alanda kullanılmakta olup bu alanların gelişiminde çok önemli bir yere sahip olduğu yadsınamamaktadır. Gözetleme sistemleri temelinde bilgisayar bilimleri kapsamındaki video ve görüntü işleme araştırma alanları bulunmaktadır. Video işleme, belirli bir video görüntüsünde var olan sahne içerisindeki değişimleri incelemede kullanılabilecek çeşitli yöntemleri içermektedir. Günümüzde video işleme bilgisayar bilimlerinin en önemli araştırma alanlarından birisidir. İki-boyutlu videolar; çoklu ortam içerik tabanlı endekslemede, bilgi elde etmede, görsel gözetleme ve dağıtık çapraz-kamera ile gözetleme sistemlerinde, insan takibi, trafik izleme ve benzeri uygulamalardaki çeşitli bölütleme, nesne tespit ve takibinde kullanılmaktadır.

1. GİRİŞ

Teknolojik gelişmeler her alanda olduğu gibi görüntü ve video işleme konularında da çığ gibi artan bir şekilde büyümektedir. Bu gelişim son 40 yıldır teknolojideki gelişmelere paralel olarak daha da artmıştır. Bilgisayarların giderek boyutlarının küçülmesi, bellek kapasitelerinin ve veri işleme hızlarının artışı görüntü işleme teknolojilerindeki gelişmeyi hızlandırmıştır. Boyutların küçülmesi ile birlikte görüntü işleme uygulamaları cep telefonlarından fotoğraf makinelerine, askeri uygulamalardan sağlık alanına birçok alanda kullanılmaktadır. Görüntülerde hareketli nesnelerin takip edilmesi bilgisayarlı görme uygulamalarındaki önemli konulardan biridir. Hareket tespiti ve analizi konusunda yapılmış çok sayıda farklı uygulamalar vardır. Örneğin askeri uygulamalar kapsamında hareketli bir hedefin takip edilerek imha edilmesi, ulusal güvenlik için akıllı silahların geliştirilmesi açısından büyük bir önem taşımaktadır. Benzer şekilde hassas güvenlikli ortamlardaki insan aktivitelerinin otomatik olarak yorumlanabilmesi, insanları algılama ve takip etme yeteneğine sahip görmeye dayalı, sağlam ve güvenilir bir sistemin kurulmasıyla sağlanabilir. Hareketli hedeflerin bulunması yol trafik kontrolü, otopark kontrolü gibi durumlarda da önem kazanmaktadır . Hareket verilerine doğru ulaşabilmek ise görüntüdeki ilgilenilen nesnenin şekil ve konum bilgilerinin minimum hatayla tespit edilmesini gerektirmektedir .Görüntüde aranan nesnenin kenarlarının doğru ve hatasız bulunması ya da cisim hareketinin hassas tespit edilmesi, cismin gerçek şeklini de ortaya çıkarmaktadır

2. MATLAB İLE NESNE TAKİBİ

Matlab görüntü işleme konusunda da diğer birçok konuda olduğu gibi oldukça gelişmiş araçlara sahip bir programdır. Bu yazıda matlab programının özellikleri kullanarak kameradan gerçek zamanlı nesne takibi yapan bir proje geliştireceğiz. Projenin özellikleri ise şu şekilde olacak;

- Gerçek zamanlı takip
- Birden fazla objenin ayırt edilebilmesi
- Görüntüde tek bir obje varsa hareketinin (engel arkasında görünmeme durumunda) tahmini
- Bulunan objelerin gösterilmesi

Bir nesneyi takip etmek için çeşitli görüntü işleme teknikleri birbiri ardına kullanılarak, görüntüler daha işlenebilir ve anlaşılabilir hale getirilmektedir. Bu kapsamda sırasıyla hangi işlemler yapacağımıza bakalım.

Bu projede nesne takibi yapılırken ilk olarak kullanım kolaylığı açısından ve avantaj sağlamak için ayrı bir görüntü alma fonksiyonu yazılarak dışardan harici bir kamera donanımı kullandığımızda da herhangi bir değer girmeden görüntü alma imkanı sağlanır. Daha sonra bir referans çerçevesi alınmaktadır. Sonraki görüntüler bu referans çerçevesi ile karşılaştırılır. Alınan çerçevedeki değişimler çeşitli teknikler kullanılarak belirlenebilir ve hareketleri öngörülür. Referans çerçevesine göre değişimler oldukça nesneler ve hareketleri algılanır. Referans çerçevesine göre değişimin olduğu yerler nesnelerin olduğu yerlere denk gelir. Nesnelerin bulunduğu yerler beyaza yakınsanır etrafı ise siyaha yakınsanır. Böylelikle nesneler o ortamdan seçilmiş olur. Median filtresi ile seçilen nesneler üzerindeki gürültüler yok edilir ve nesne daha sağlıklı bir şekilde belirlenir. Nesneyi çevreleyen çerçeve ile ana ekran karşılaştırılır ve nesnelerin merkez noktası bulunarak ekranın neresinde olduğu tespit edilir. Ve daha sonra tespit edilen bu noktanın koordinatları ekrana yansıtılarak kullanıcıya sarı işaretçilerle gösterilir.

Öncelikle kameradan aşağıdaki gibi bir görüntü elde ettiğimizi varsayalım ve buradan mavi renkli objeleri takip etmek istediğimizi düşünelim.

2.1 Görüntü Alma ve Ayarlar

Video işleme işleminde iyi sonuçlar alınabilmesi için görüntünün düzgün bir şekilde alınabilmesi çok önemlidir ve diğer adımları tamamı ile etkilemektedir. Görüntünün hangi aygıtla alındığı ve hangi çözünürlükte olduğunun belirlenmesi için imaq.VideoDevice komutu kullanılmaktadır. Bu komut, görüntülerin tek bir kare şeklinde alınabilmesini sağladığı için işlem yapılırken kolaylık sağlamaktadır. Aynı zamanda da farklı aygıtlardan görüntü alımını sağladığı için ve aygıta özgü sayısal olarak skaler değerler verebildiği için kullanıcıya kolaylık sağlamaktadır. Ayrıca bu projede kullanım kolaylığı ve avantaj sağlamak için görüntü almak için ayrıca bir cameraInfo.m fonksiyonu kullanılarak ayrı bir kamera kullanılması halinde programın kameranın özelliklerini tanıyıp görüntü alması sağlanmıştır. imaqhwinfo komutu görüntüleyici araç ile matlab haberleşmesini kuran ve operatörün görüntüyü görmesini sağlayan arayüzdür. Bu iki komut ile hangi cihazın seçildiği, hangi çözünürlükte ekrana verildiği ortaya konulmuştur. Görüntü alma işlemleri için oluşturulan fonksiyon ,yazılan kod ve kodun çıktısı olarak ekrana gelen görüntü gösterilmiştir.

camera... ×

```
function [camera_name, camera_id, resolution] = cameraInfo(hardwareInfo)
    % Donanım bilgisine göre kameranın isminin, bilgilerinin, çözünürlük
    % değerlerinin alınması.
    camera_name = char(hardwareInfo.InstalledAdaptors(end));
    camera_info = imaqhwinfo(camera_name);
    camera_id = camera_info.DeviceInfo.DeviceID(end);
    resolution = char(camera_info.DeviceInfo.SupportedFormats(end-2));
    % Burada çözünürlük değerlerinden sondan 2 önceki değer seçiliyor.
    % Çünkü kullandığımız kamerada istenen çözünürlük (800x600) bu değere denk geliyor.
end
```

1. şekil: Görüntü almak için kullanılan fonksiyon

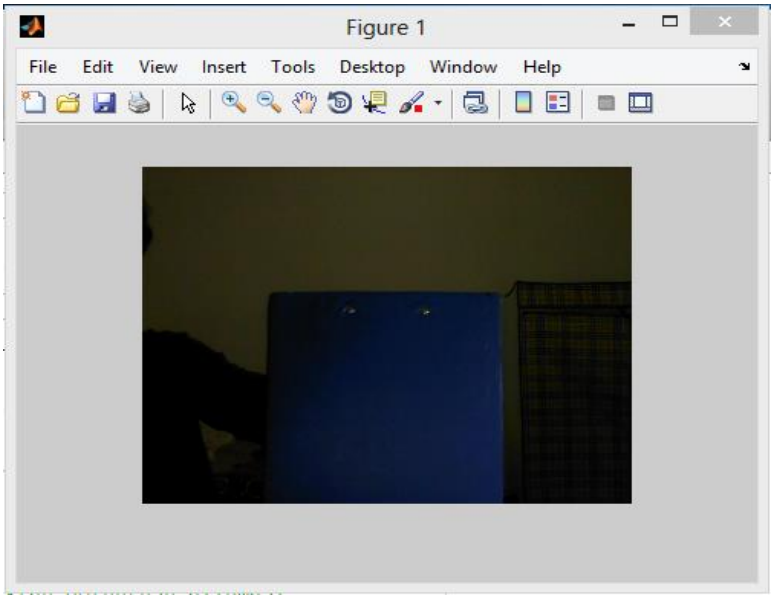
ekrang... ×

```
hardwareInfo=imaqhwinfo; % Kamera donanımı hakkında gerekli bilgilerin alınması
[camera_name, camera_id, format]=cameraInfo(hardwareInfo); % Donanım bilgilerinden
%kameranın özelliklerinin ayıklanması
memoryInfo=imaqmem; % Bilgisayarın bellek bilgisinin alınması
imaqmem(memoryInfo.AvailPhys); % Kullanıma müsait bütün belleğin görüntü
%işleme işlemleri için kullanılabilir hale getirilmesi
video = videoinput(camera_name, camera_id, format); % Kamera özelliklerine göre
%video objesinin oluşturulması

set(video, 'FramesPerTrigger', Inf); % Videonun sürekli frame alması için gerekli düzeltme
set(video, 'ReturnedColorspace', 'rgb'); % Videonun rgb uzayda dönmesi için gerekli ayarlama
video.FrameGrabInterval = 1; % Videodan ne kadar sıklıkla frame çekileceği.

start(video); % Videonun başlatılması
 hmain = gca;
preview(video);
data = getsnapshot(video); % Videodan ekran görüntüsü alınması
imshow(data,'Parent',hmain); % Mavi objelerin boyanmış halde gösterilmesi
stop (video);
```

2.şekil : Videonun başlaması için gereken kod ve ayarlamalar



3.şekil : Programın çıktısı

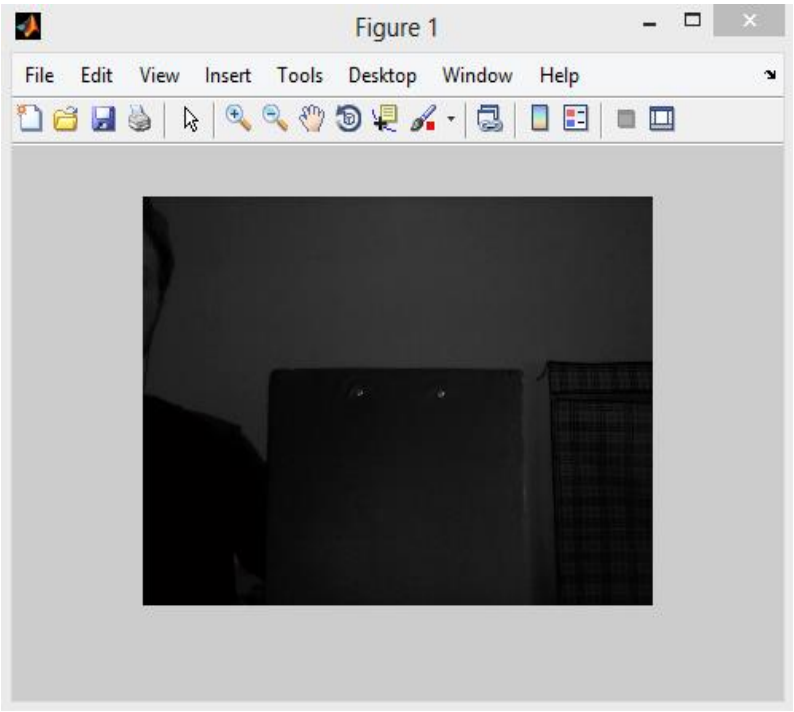
2.2 Nesne Algılama

Nesne algılanıp sınırları belirlenerek çerçevelendikten sonra istenilen fonksiyonların yapılabilmesi ve operatörün ayarlamaları kolaylıkla yapabilmesi açısından hesaplanan değerler ekrana yazılması sağlanmaktadır. Nesne algılanıp sınırları belirlenerek çerçevelendikten sonra istenilen fonksiyonların yapılabilmesi ve operatörün ayarlamaları kolaylıkla yapabilmesi açısından hesaplanan değerler ekrana yazılması sağlanmaktadır.

Buradan mavi renkleri ayıklamak için birkaç adım gereklidir. Öncelikle 3 renk katmanından (RGB) oluşan bu resmin mavi katmanını ayıklamak ve gri katmana göre farkına bakmak bize oldukça iyi bir tahmin verecektir. Resmin mavi katmanı aşağıdaki gibidir. Bu katman data değişkeninin 3. boyutunda tutulmaktadır ve erişmek için **data(:,:,3)** yazılması yeterlidir. Resmin gri renkli hali için ise Matlab içinde bulunan **rgb2gray()** fonksiyonu kullanılabilir.

```
20 -    preview(video);
21 -    data = getsnapshot(video); % Videodan ekran görüntüsü alınması
22 -        img =rgb2gray( data );
23 -        imshow(img,'Parent',hmain); % resmin gri hali
24 -    stop (video);
25
```

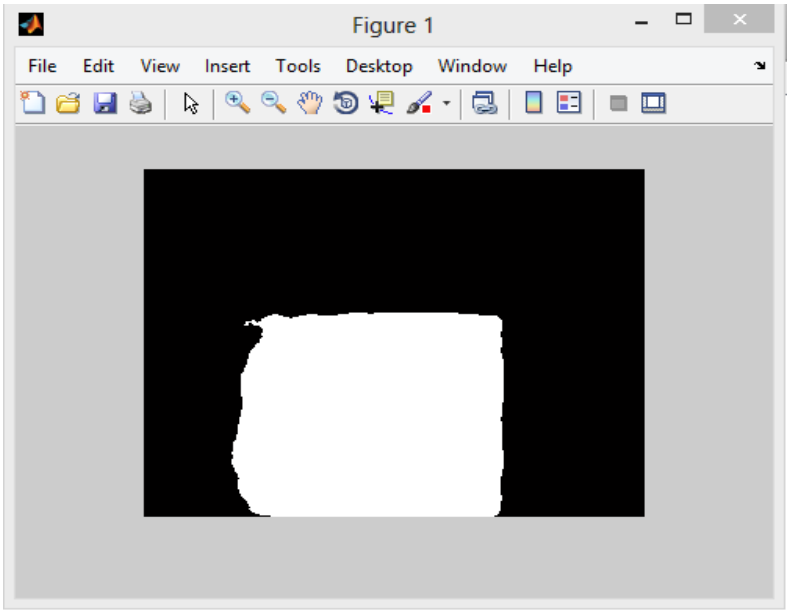
4. şekil : Ekran görüntüsünün gri hali



5.şekil: Program çıktısı

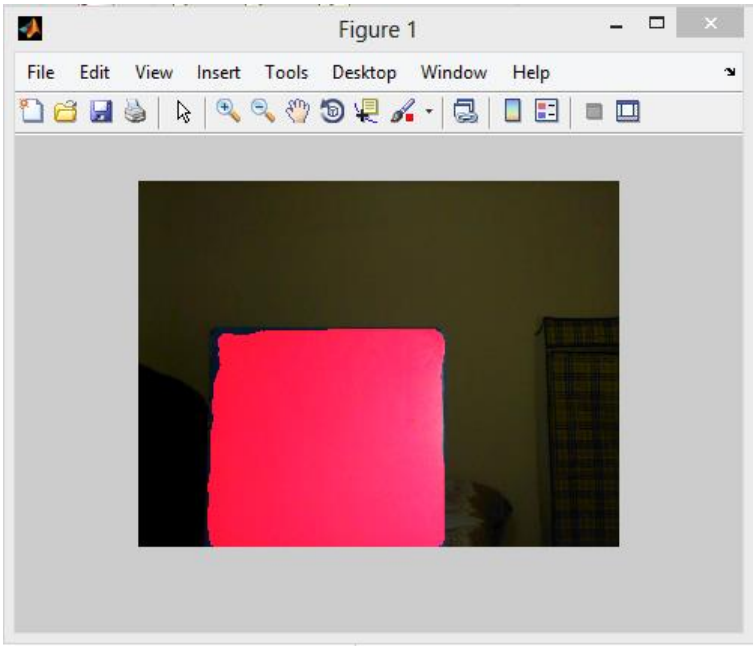
Daha sonra bulunan mavi ve gri katmanlar birbirinden **imsubtract()** fonksiyonu ile çıkartılıp, aşağıdaki fark elde edilir. Bu resme "**Median Filtresi**" uygulanarak tuz-biber gürültüsü adı verilen karıncalanmaların giderilmesi sağlanabilir. Görüntü veya sinyal işlenirken gürültü olmaması önemlidir. Bunun içindir ki gürültüyü engelleyen doğrusal olmayan sayısal filtreme tekniği olan median filtresi kullanılmaktadır.
Bu filtreleme işlemi görüntü işlemenin daha sonraki adımlarında daha net değerler alabilmek için bir ön işleme anlamına da gelmektedir. Görüntüsü işlenen maddenin gürültüden arındırılırken genel özellikleri bozulmadığı için (madde kenarları gibi) median filtresi sayısal görüntü işleme de yaygın olarak kullanılmaktadır. Bu işlem 2 boyutlu median filtresi fonksiyonu olan **medfilt2()** fonksiyonu ile yapılabilir. Daha sonra eşik değerine göre resim içindeki renkler ayırt edilip, resim daha kolay çalışılabilir hale getirilmiştir. Bu işlemler şöyle yapılmaktadır.

```
14 -    start(video); % Videonun başlatılması
15      % Çözünürlüğün dinamik olarak ayarlanması
16 -    resolutionY = str2double(format(strfind(format,'_')+1:strfind(format,'x')-1));
17 -    resolutionX = str2double(format(strfind(format,'x')+1:length(format)));
18      % Değişkenlerin ön tanımlanması
19 -    hmain = gca;
20 -    img = zeros(resolutionX,resolutionY);
21 -    imgSize = resolutionX*resolutionY;
22 -    threshold = 33; % Mavi renk katmanından ne kadar mavi olan renklerin
23      %seçileceğinin eşik değeri
24 -    preview(video);
25 -    data = getsnapshot(video); % Videodan ekran görüntüsü alınması
26 -        img =rgb2gray( data );
27 -        img = imsubtract(data(:,:,3), rgb2gray(data)); % Mavi resim katmanından gri
28          % katmanın çıkarılması
29 -        img = medfilt2(img, [9 9]); % 9x9 Median filtresi uygulanması
30          % Eşik değerine göre renklerin ayıklanması
31 -        img(img<threshold) = 0;
32 -        img(img>=threshold) = 255;
33 -        img = bwareaopen(img,(imgSize)/1200); % Küçük objelerin resimden ayıklanması
34          % (Resim alanın 1200'de birinden küçük objeler atılmaktadır..
35 -        imshow(img,'Parent',hmain); % Mavi objelerin boyanmış halde gösterilmesi
36 -    stop (video);
```



6.şekil: Mavi nesnenin algılanması için mavi ve gri katmanın farkı ve median filtresi sonucu

Daha sonra bwareaopen komutu ile belli bir büyüklüğün altındaki küçük objelerin resimden ayıklanması (Resim alanın 1200'de birinden küçük objeler atılmaktadır) sağlandıktan sonra resme biraz makyaj yapmak için `data(img>0) = 255;` satırı eklenerek mavi nesneler üzerine makyaj yapılarak belirginleştirilmesi sağlanmıştır. Bu işlemlerin sonucunda elde edilen ekran görüntüsünün son hali şöyledir;



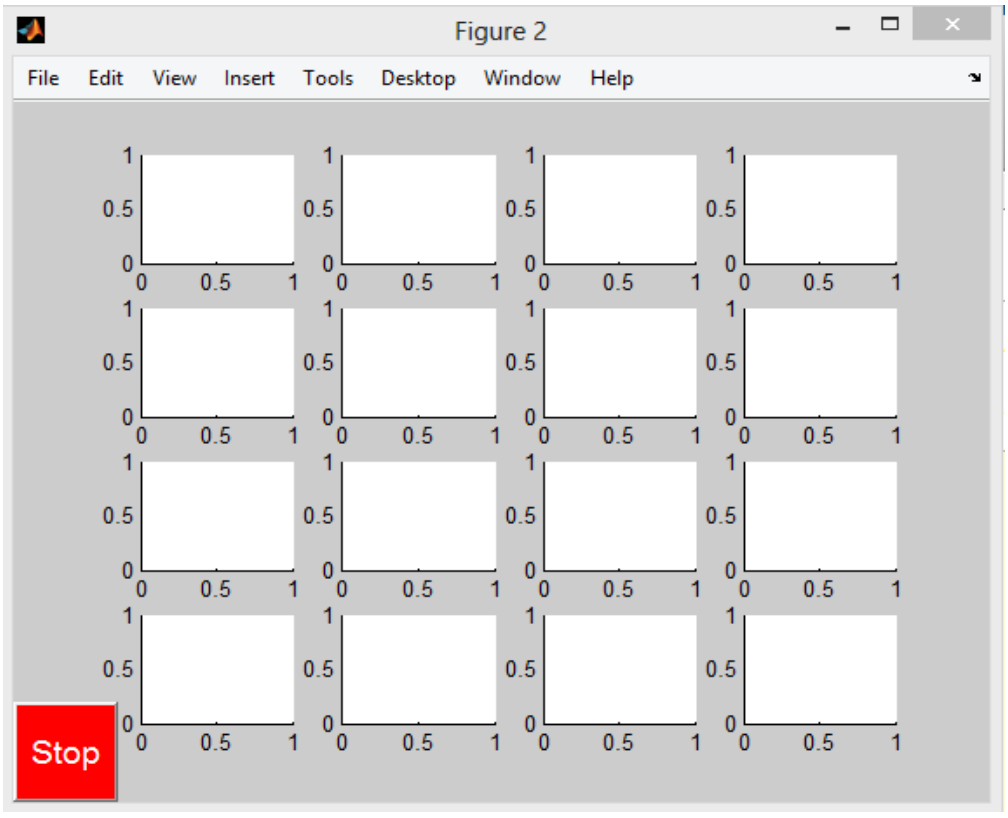
7.şekil : Makyajlama işlemi sonucu (dilation)

2.3 Stop Butonu Eklenmesi Ve Algılanan Nesnelerin Gösterilmesi

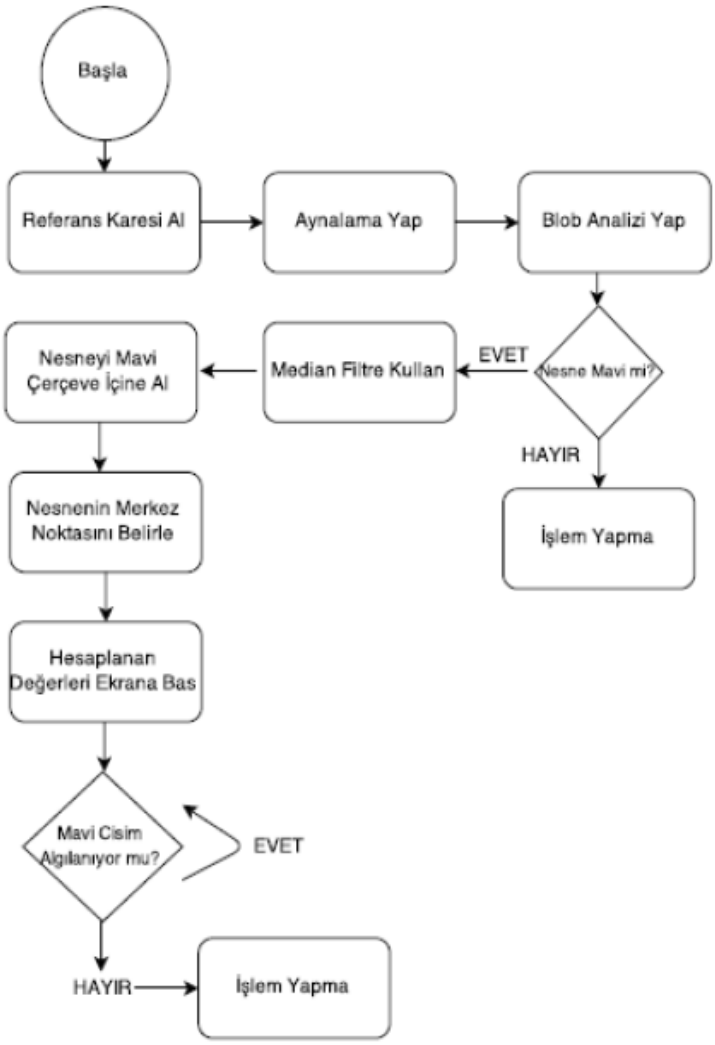
Tüm bu işlemler yapılırken ve aşamalar birbiri üzerine eklenirken kolaylık sağlaması için kod çalışır durumdayken üzerine basıldığında videoyu ve aynı zamanda kodun işleyişini durduran ve daha sonra 'Run' komutu ile kaldığı yerden devam etmemiz sağlayan bir stop butonu eklenmiştir. Videonun arka planda çalışır halde kalmaması için "stop" düğmesinin kullanılması tavsiye edilmektedir. Aksi halde el ile video objesinin durdurulması gerekmektedir. Video objesi kullanım halindeyken ikinci kez çalıştırılması veya video objesinin silinmesi hataya neden olmaktadır. Bu durumda Matlab'ın yeniden başlatılması gerekebilir. Durdurulan kod yeniden çalıştırılmak istenirse stop düğmesiyle durdurulması durumunda kısayol olarak F5 ile yeniden çalıştırılabilir.

Ayrıca ek bir figür penceresi açılarak videoda algılanan nesnelerin maksimum sayısı ve bu algılanan nesnelerin figüre penceresi üzerine kaydedilmesi sağlanmıştır. Tüm bu işlemlerin sonucu açılan ek figür penceresi ve stop butonu için gerekli olan kodlar ve çıktısı aşağıda gösterilmektedir;

```
%% Videonun ve bulunan objelerin gösterilmesi içinfigure'lerin açılması
figure;
hmain = gca;
f=figure;
% Durdurma Butonu
isRunning = 1;
u=uicontrol('String','Stop','Callback',...
    'isRunning = 0; disp(''Hesaplamalar Durduruldu.'')',...
'ForegroundColor','w','BackgroundColor','r',...
'Fontsize',14,'FontWeight','Demi','Position',[1 1 60 60]);
set(u,'position',[1 1 60 60])
%stop butonunun görünümü ile ilgili yazı ve
%butonun büyüklük ve pozisyon ayarları
hsub = zeros(1,16);           %bulunan ve gösterilecek olan maksimium nesne
                              % sayısının belirlenmesi
for i = 1:16                  %ve gerekli figure pencerelerinin açılması
     hsub(i) = subplot(4,4,i);
end
```



2.4 Nesne Algılama İçin Blok Diyagram



2.5 Algılanan Nesne İçin Hesaplamalar

Algılanan mavi nesnenin özelliklerini bilmek ve nesnenin bu özelliklerinden oluşan değerlerinin hesaplanması projemizde önemli bir yer tutmaktadır. Değerlerin çıktı ekranında görüntülenmesi konusunda da anlatıldığı gibi referans noktasına olan uzaklık, çerçeve alanı, nesnenin orijine olan uzaklığı ve nesnenin koordinatları aşağıda yapılan işlemler sonucunda hesaplanmaktadır.

Burada yapılan işlemleri kısaca özetlemek gerekirse, öncelikle **regionprops()** fonksiyonu ile bağlı bileşenlerin ve bu bileşenlerin istenen özelliklerinin elde edilmesi sağlanmaktadır. 4 komşuluğuna göre bağlı bileşenlerin bulunması 8 komşuluğu ile aynı sonucu daha hızlı sağlamakta olduğundan 4 komşuluğuna göre bağlı bileşenlerin özellikleri çıkarılmıştır. Bunları sağlayan kod aşağıdadır.

```
[Label, Count] = bwlabel(img, 4); % 4 Komşuluğuna göre bağlı
% bileşenlerin bulunması (8 komşuluğu ile aynı sonucu daha hızlı sağlamakta)
clc
disp(['Count = ' num2str(Count)]);
stats = regionprops(logical(Label),'BoundingBox','Centroid','FilledImage','Area');
% Bağlı bileşenlerin özelliklerinin çıkarılması
```

Daha sonra bulunan bu özelliklere göre her bir objenin alanını çevreleyen bir sınır çizilip, merkez noktası belirlenerek bu noktanın yanında cisimle ilgili bilgiler gösterilmektedir. Bu bilgiler cismin merkezinin koordinatları, hareketli cismin x ve y eksenlerinde sahip olduğu hız ve ayrıca cismin alanı bilgilerini içermektedir. Bu işlemlerin sonucu aşağıdaki gibidir.

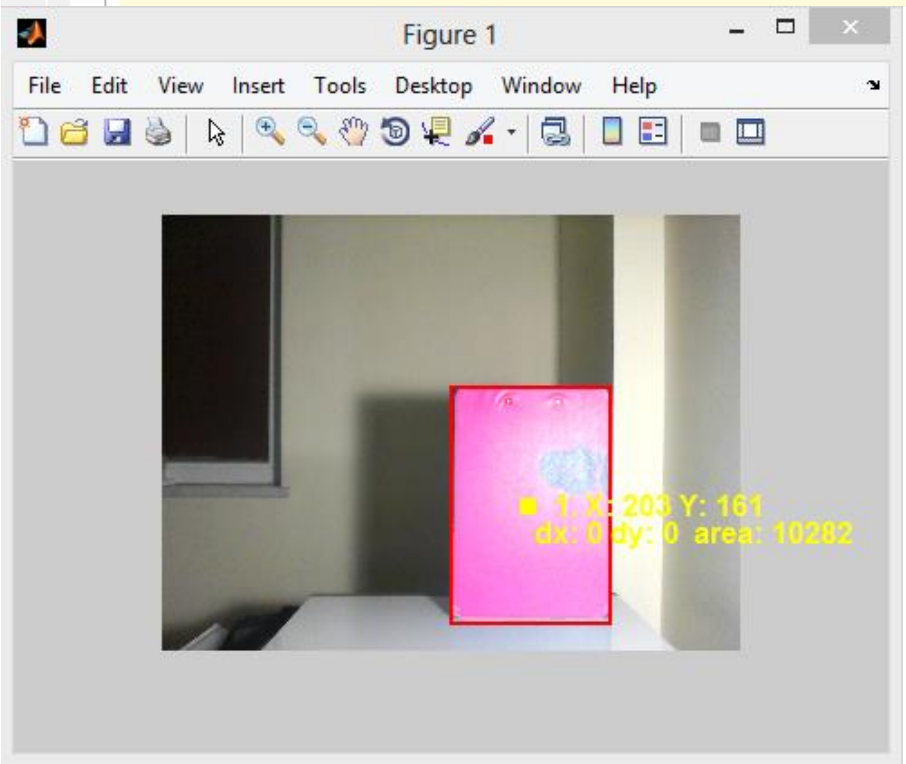
```
if(exist('stats','var')) % Herhangi bir obje bulunması durumunda
  for object = 1:length(stats) % Bulunan her obje için
     if(object < 16 && object > 0) % Obje sayısı sınırlaması
     boundingBox{object} = stats(object).BoundingBox; % Objeyi çevreleyen
                                                      % dikdörtgeninkoordinatları
     centroid{object} = stats(object).Centroid; % Objenin merkez noktası
                                                % koordinatları
     area(object,2) = stats(object).Area; % Objenin alanı
       if(area(object,2) >= 0)     % Objenin alanının kontrolüne göre önceki
                                   %alan bilgisinin güncellenmesi
         area(object,1) = area(object,2);
       end
```

```
     % Objenin bir önceki konumundan ne kadar uzaklaştığının hesaplanması
     dx=round(objPosition(2,1,object)-objPosition(1,1,object));
     dy=round(objPosition(2,2,object)-objPosition(1,2,object));

     filledImage{object} = stats(object).FilledImage; %Objenin küçük
                                                      %resminin alınması
     rectangle('Position',boundingBox{object},...
         'EdgeColor','r','LineWidth',2,'Parent',hmain)  %Objenin etrafına
                                                        %dikdörtgen çizilmesi
     plot(hmain,centroid{object}(1),centroid{object}(2),...
         ' ys','MarkerSize',6,'MarkerFaceColor','y')      %Merkez noktasına
                                                          %sarı işaretçi konulması
     imshow(filledImage{object},'Parent',hsub(object)) % Bulunan küçük resimlerin
                                                       %ikinci grafiğe çizilmesi
     title(hsub(object),[num2str(object) '. Obje']); %Objelerin başlıklarının
                                                     %atanması
     objPosition(1,:,object) = objPosition(2,:,object); % Objenin eski
                                                        % konumunun güncellenmesi
     objPosition(2,:,object) = [centroid{object}(1),centroid{object}(2),...
         area(object,2)/imgSize]; % Objenin yeni konumunun bulunması
```

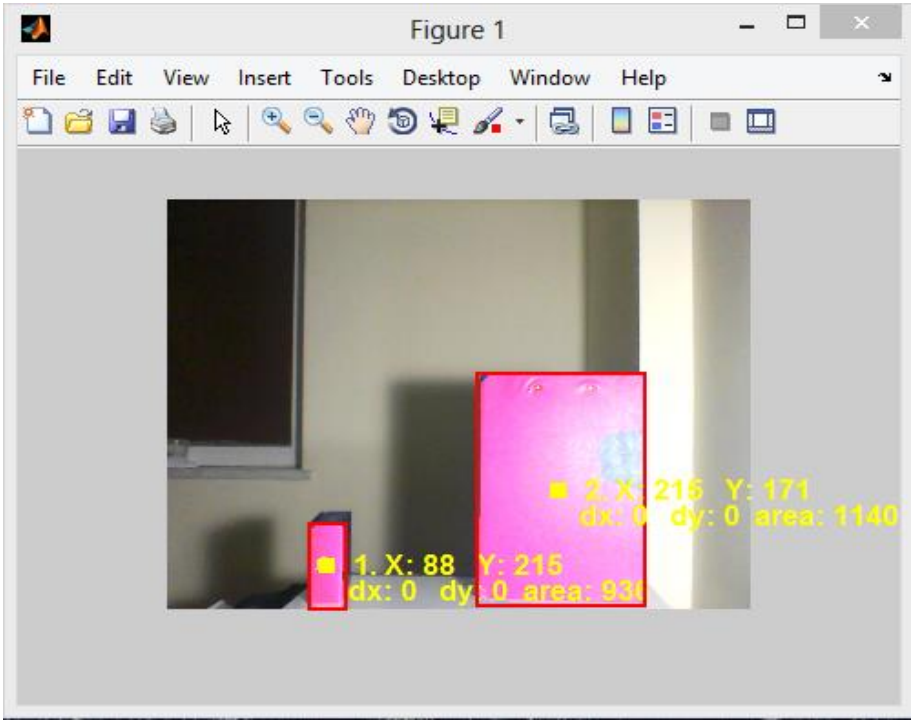
Tüm bu bilgiler sağlandıktan sonra videoda tek bir nesne algılandığı durumda bu nesnenin hız, konum ve alan bilgilerinin elde edilmesini ve ekrana yazılmasını sağlayan kod ve çıktısı aşağıdaki gibi olmaktadır;

```
95 -    if(length(stats) == 1) % Tek obje bulunması durumunda
96          % Objenin konumlarının bulunmasın ve tek obje olması
97          % durumunda hız ,konum ,alan bilgileri
98
99 -          x = boundingBox{object}(1);
100 -         y = boundingBox{object}(2);
101
102 -            txtInfo = text(centroid{object}(1)+15,centroid{object}(2),...
103             [num2str(object) '. X: ' num2str(round(centroid{object}(1)))...
104             ' Y: ' num2str(round(centroid{object}(2)))],'Parent',hmain);
105 -           txtSpeed = text(centroid{object}(1),centroid{object}(2)+15,...
106             [' dx: ' num2str(dx) ' dy: ' num2str(dy) '  area: '...
107             num2str(round(area(object,2)))],'Parent',hmain);
108 -                   set(txtInfo, 'FontName', 'Arial', 'FontWeight', 'bold',...
109                         'FontSize', 12, 'Color', 'yellow');
110 -                   set(txtSpeed, 'FontName', 'Arial', 'FontWeight', 'bold',...
111                         'FontSize', 12, 'Color', 'yellow');
112                     % Gürültüden veya objenin engel arkasına gitmesindendolayı
113                     % alandaki değişikliklerin ekarte edilmesi
114                     % bu kısım tek obje olması durumunda hesaplanabileceğinden
115                     % if else kullanılarak ayrıca belirtildi
116
117 -                   if(area(object,2)/area(object,1) >= 0.9 &&...
118                             area(object,2)/area(object,1) <= 1.1)
119 -                       theImage = stats(object).FilledImage;
120 -                   end
```



Eğer videoda birden fazla mavi nesne algılanırsa aşağıdaki kod çalışır ve kaç tane nesne varsa hepsinin konumu, merkez koordinatları, hızı ve alan bilgileri kullanıcıya cismin yanında gösterilir..

```
117 -   else % Birden fazla obje olması durumunda
118         % Hız, Konum, Alan bilgilerinin kullanıcıya gösterilmesi
119 -       txtInfo = text(centroid{object}(1)+15,centroid{object}(2), [num2str(object)...
120         '. X: ' num2str(round(centroid{object}(1)))...
121         ' Y: ' num2str(round(centroid{object}(2)))],'Parent',hmain);
122 -       txtSpeed = text(centroid{object}(1),centroid{object}(2)+15,...
123             [' dx: ' num2str(dx)...
124              ' dy: ' num2str(dy) '  area: ' num2str(round(area(object,2)))],'Parent',hmain);
125 -       set(txtInfo, 'FontName', 'Arial', 'FontWeight', ...
126             'bold', 'FontSize', 12, 'Color', 'yellow');
127 -       set(txtSpeed, 'FontName', 'Arial', 'FontWeight',...
128             'bold', 'FontSize', 12, 'Color', 'yellow');
129 -   end
```



Sahnede tek bir obje olması durumunda eğer obje bir engelin arkasına giriyorsa çalışan kod şu aşağıdadır. Burada objenin bilgileri var ancak ekranda hiç obje yoksa cismin sabit hızla ve aynı yönde gittiğini kabul edersek harket tahmini;

```
134         % Objenin bilgileri varsa fakat ekranda hiç obje yoksa cismin
135         % hareketinin tahmini
136 - if(exist('object','var') && exist('x','var') && exist('y','var') && length(stats) < 1)
137             % Cismin sabit hızla aynı yönde gittiği tahmin edilerek bir sonraki
138             % konumunun hesaplanması
139 -           x=x+dx;
140 -           y=y+dy;
141             % Bulunan konuma küçük resmin çizilmesi
142 -           image(x,y,theImage*255,'Parent',hmain);
143 - end
```

Son olarak ise performans sağlamak için alınan verilerin bellekten silinmesi işlemi vardır.

```
148         % Performans açısından RAM'in dolmaması için belleğin her 100 kare
149         % alındığında bir sıfırlanması
150 -       if mod(video.FramesAcquired,100)==0
151 -           flushdata(video);
152 -       end
153 -   end
154
155     % Stop düğmesine basıldıktan sonra döngünden çıkıp videoyu durdurup
156     % belleğin boşaltılması
157 -   stop(video);
158 -   flushdata(video);
```

3.Sonuç

Bu projede nesne takibi yapılırken ilk olarak kullanım kolaylığı açısından ve avantaj sağlamak için ayrı bir görüntü alma fonksiyonu yazılarak dışardan harici bir kamera donanımı kullandığımızda da herhangi bir değer girmeden görüntü alma imkanı sağlanır. Daha sonra bu görüntü alındıktan sonra video işleme aşamalarına geçilmiştir.

Videonun ekran görüntüsü alındıktan sonra mavi renkleri ayıklamak için birkaç adım gereklidir. Öncelikle 3 renk katmanından (RGB) oluşan bu resmin mavi katmanını ayıklamak ve gri katmana göre farkına bakmak bize oldukça iyi bir tahmin verecektir. Resmin mavi katmanı data değişkeninin 3. boyutunda tutulmaktadır ve erişmek için **data(:,:,3)** yazılması yeterlidir. Resmin gri renkli hali için ise Matlab içinde bulunan **rgb2gray()** fonksiyonu kullanılabilir.

Daha sonra bulunan mavi ve gri katmanlar birbirinden **imsubtract()** fonksiyonu ile çıkartılıp, bir fark değeri elde edilir. Bu resme "**Median Filtresi**" uygulanarak tuz-biber gürültüsü adı verilen karıncalanmaların giderilmesi sağlanabilir. Daha sonra videodaki nesneye makyaj ve kenar bulma işlemleri sırasıyla uygulanarak cismin tespiti işlemi ile devam edilir. Açılan ek bir figür penceresine bulunan tüm nesnelerin bilgilerinin kaydedilip kullanıcıya gösterilmesi işlemi ile sonlandırılır..

Tüm bu işlemler sonrasında algılanan nesne için videodaki hareketine göre bilgilerinin alınması işlemi gelmektedir. Burada cisim algılandıktan sonra merkez noktası belirlenir ve sarı bir işaretçi konularak bu merkez videoda gösterilir. Daha sonra cismin X ve Y düzlemindeki konumları hesaplanır ve değerler ekrana yazdırılır. Eğer cisimler ya da cisim hareket halindeyse dx ve dy değişkenleri ile x ve y eksenlerindeki hızlarının değişimi de aynı şekilde ekrana yazdırılmaktadır.

Daha sonra cismin alan hesaplamaları yapılarak belli bir alan değerinin altındaki cisimlerin obje olarak algılanmaması sağlanmaktadır. Ve yine alan hesaplamalarına bağlı olarak cismin ekranda herhangi bir engel arkasında kalması durumları için alan değişimlerinin ekarte edilmesi işlemi gerçekleştirilmiştir. Ayrıca videoya daha önce girmiş bir nesnenin bilgileri varsa ancak obje sonraki durum için ekranda değilse bu nesnenin sabit hızla ve aynı yönde gittiği varsayılarak hareketinin tespiti için de ayrıca algoritma geliştirilmiştir

Ayrıca tüm bu işlemler yapılırken ve yapıldıktan sonra kullanım kolaylığı olması açısından videonun durdurulup tekrar çalışması için ‘’stop’’ butonu eklenmiştir. Videonun arka planda çalışır halde kalmaması için "stop" düğmesinin kullanılması tavsiye edilmektedir. Aksi halde el ile video objesinin durdurulması gerekmektedir. Video objesi kullanım halindeyken ikinci kez çalıştırılması veya video objesinin silinmesi hataya neden olmaktadır. Bu durumda Matlab'ın yeniden başlatılması gerekebilir. Durdurulan kod yeniden çalıştırılmak istenirse stop düğmesiyle durdurulması durumunda kısayol olarak ‘’run’’ ile yeniden çalıştırılabilir.