IMAGE PROCESSING (MSE4631)

OBJECT TRACKING USING COLOR FEATURE, FRAME DIFFERENCING AND BLOB ANALYSIS

Sara Hesham 180103

Dima Shaltaf 182717

Yahia Emad 183861

Contents

Object tracking using Colour Feature, Frame Differencing and Blob Analysis.

Abstract

Detecting and tracking objects are among the most prevalent and challenging tasks that a surveillance system has to accomplish in order to determine meaningful events and suspicious activities, and automatically annotate and retrieve video content. Under the business intelligence notion, an object can be a face, a head, a human, a queue of people, a crowd as well as a product on an assembly line. In this chapter we introduce the reader to main trends and provide taxonomy of popular methods to give an insight to underlying ideas as well as to show their limitations in the hopes of facilitating integration of object detection and tracking for more effective business oriented video analytics.

Keywords

object detection, object classification, object tracking, video surveillance, frame difference.

Introduction

Object tracking is the process of locating moving objects over time using the camera in video sequences. The objective of object tracking is to associate target objects in consecutive video frames. To detect or locate the moving object in frame, Object detection is first stage in tracking. After that, detected object can be classified as vehicles, human, swaying tree, birds and other moving objects. It is challenging or difficult task in the image processing to track the objects into consecutive frames. Various challenges can arise due to complex object motion, irregular shape of object, occlusion of object to object and object to scene and real time processing requirements. Object tracking has a variety of uses, some of which are: surveillance and security, traffic monitoring, video communication, robot vision and animation. This project presents the object tracking using color features, frame differencing and blob analysis.

Imagine waiting for your turn in a shopping line at a busy department store. Your visual system can easily sense humans and identify different layers of their interactions. As with other tasks that our brain does effortlessly, visual analytics has turned long out to be entangled for machines. Not surprisingly, this is also an open problem for visual perception. The main challenge is the problem of variability. A visual detection and tracking system needs to generalize across huge variations in object appearance such due for instance to viewpoint, pose, facial expressions, lighting conditions, imaging quality or occlusions while maintaining specificity to not claim everything it sees are objects of interest.

Motivation

Object detection is used in a lot of applications, this project will help us examine the possibilities of image processing which we can later use with other fields like Robotics and

Autonomous Cars.

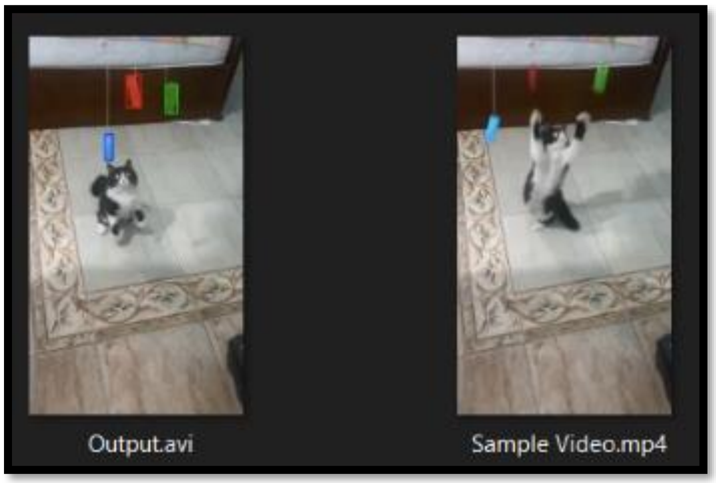
Methodology

The project will be divided into sub-processes, starting with Video Acquisition and Frame extraction. Then, each frame will be converted to grayscale then compared to the original frame's colour channel to extract the colour component of the frame. A median filter is then used to eliminate noise after the differencing process. Subsequentially, Thresholding is then used to binarize the frame, after acquiring a binary version of the frame, Blob Analysis is used to identify the location of the centroids of the object. Finally, the location markers and bounding boxes of the detected objects are drawn on the original frame and then extracted.

Video Acquisition

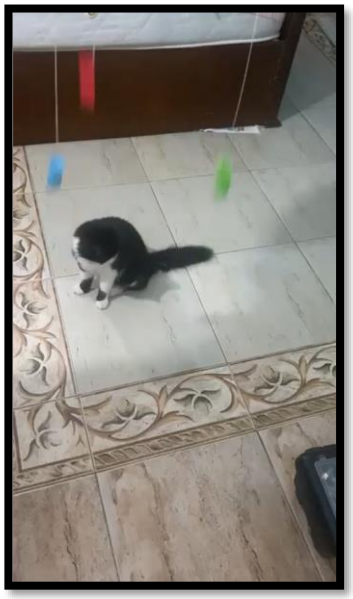
The video is acquired by recording a sample video with 3 RGB-colored moving objects and then read into MATLAB for frame extraction. Having Acquired the video, the number of frames is extracted, and an output video object is initialized for the output video later.

```
Video=VideoReader('Sample Video.mp4');
Frames=Video.NumberOfFrames;
v = VideoWriter('Output.avi');
open(v)
```



Frame Extraction

In this process, each frame of the “Frames” variable which resamples the number of frames of the whole video is extracted for processing.



```
for Frame = 1:1: Frames
    %%Extracting a frame
    CurrentFrame = read(Video,Frame);
    imshow(CurrentFrame)
end
```

Gray Scale

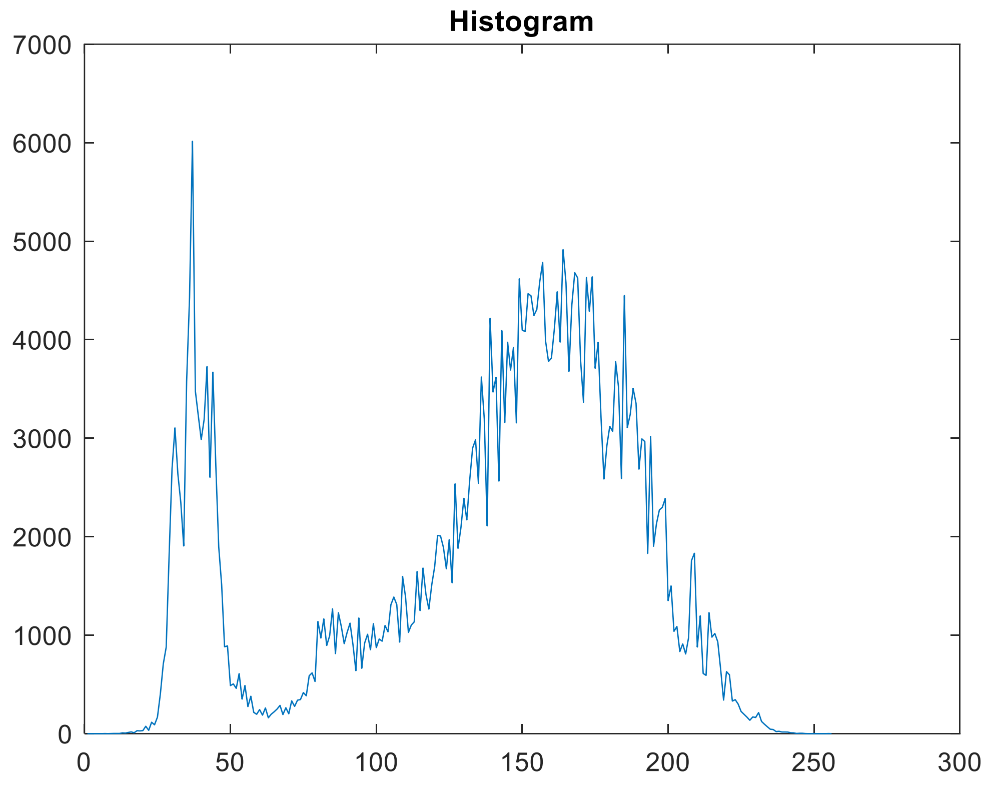
The gray scale version of the image is converted using the Weighted Method using the following equation: **Grayscale** = 0.299R + 0.587G + 0.114B

```
    Size = size(CurrentFrame);
    %%GrayScale conversion, GS_CF = GrayScale Current Frame
    GS_CF=zeros(Size(1),Size(2));
    for row = 1:Size(1)
        for column=1:Size(2)
            GS_CF(row,column)=(CurrentFrame(row,col-
umn,1)*0.2989) + (CurrentFrame(row,column,2)*0.5870) + (Cur-
rentFrame(row,column,3)*0.1140);
        end
    end
    GS_CF=uint8(GS_CF);
    %%imshow(GS_CF);
```

Histogram Analysis

To decide whether the image needs to undergo histogram equalization or not we analyzed some histograms of the frames which were all relatively similar.

```
        %%Histogram of grayscale frame
        freq=zeros(256,1);
    Pixels = Size(1)*Size(2);
        %%Extract Histogram
        for i=1:Size(1)
            for j=1:Size(2)
                value=GS_CF(i,j);
                freq(value+1)=freq(value+1)+1;
            end
        end
```

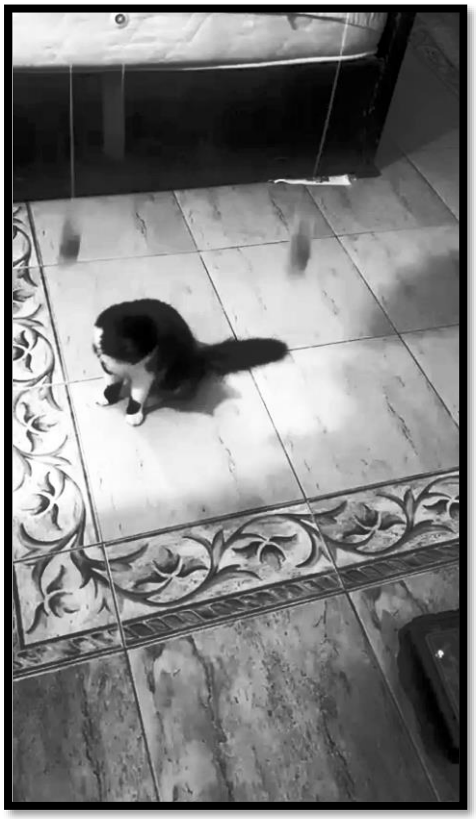


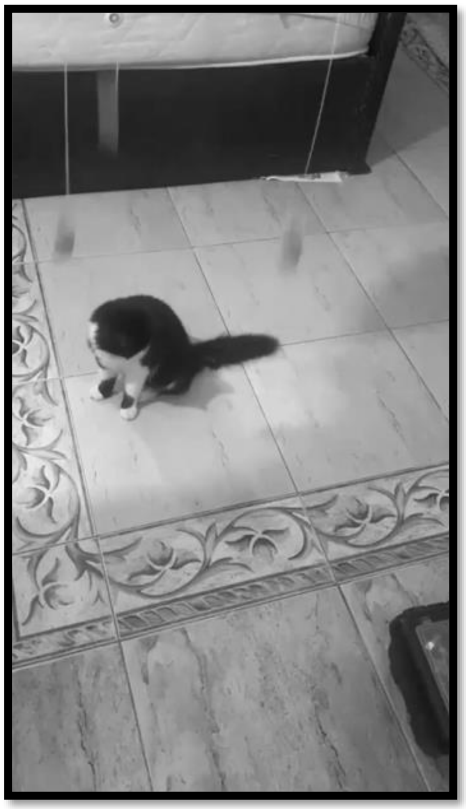
After analyzing the histogram, we decided that it needed equalizing which was also supported by examining the result of the frames after applying frame differencing where shadows were present.

Histogram Equalization

To do the equalization, process some parameters are first created to store the calculated values needed for the process.

```
    CDF=zeros(256,1);
    CDC=zeros(256,1);
    cum=zeros(256,1);
    output=zeros(256,1);
%%Calculate CDF
    for i=1:Size(1)
        for j=1:Size(2)
            CDF(value+1)=freq(value+1)/Pixels;
        end
    end
    sum=0;
    res=255;
    %Calculate CDC
    for i=1:size(CDF)
        sum=sum+freq(i);
        cum(i)=sum;
        CDC(i)=cum(i)/Pixels;
        output(i)=round(CDC(i)*res);
    end
    %%Applying new histogram to the image
    HE_CF=uint8(GS_CF);
    for i=1:Size(1)
        for j=1:Size(2)
            HE_CF(i,j)=output(GS_CF(i,j)+1);
        end
    end
    %%imshow(HE_CF)
        figure,plot(output); title('Histogram');
```

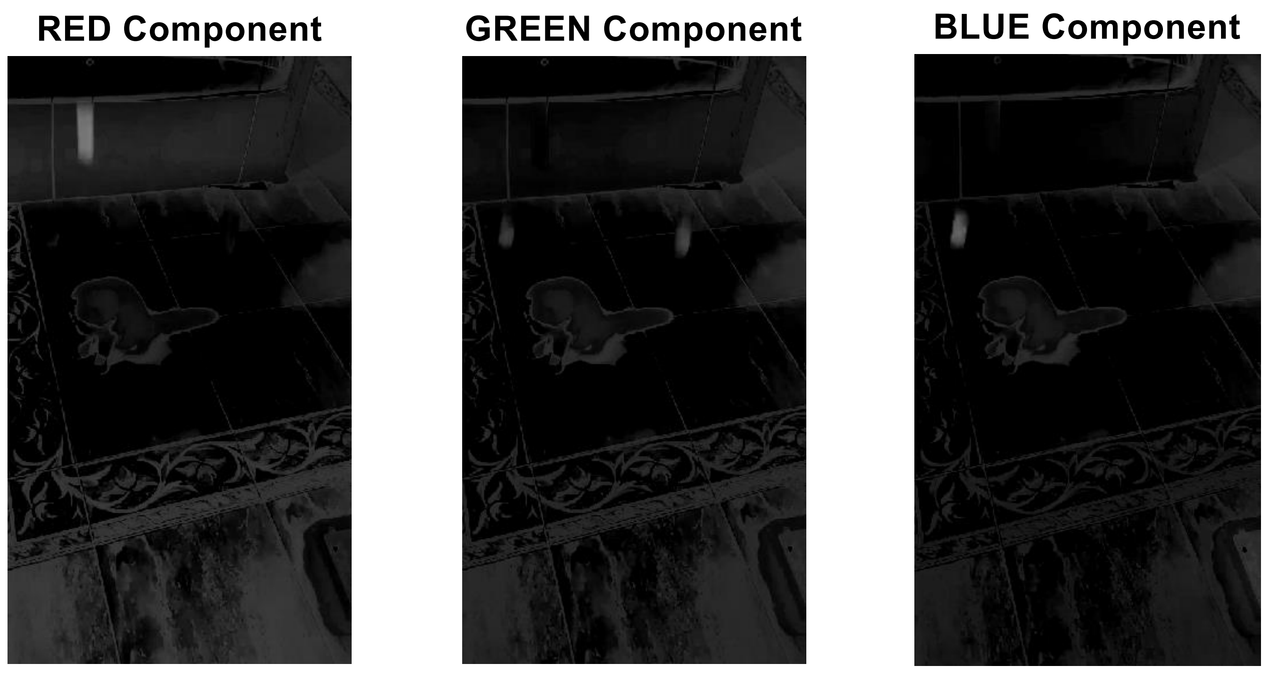

Histogram Equalized


Original

Frame Differencing

The first step to start detecting our objects is by applying frame differencing techniques on the three RGB layers of the frame to extract the red, green, and blue components of the image respectively. We take each layer and subtract the gray scale version of the frame from it, the result is the colored object corresponding to the layer being subtracted from. For example, if we take the red layer and subtract the gray scale version, we remove the background and keep any red colored objects in the resultant image.

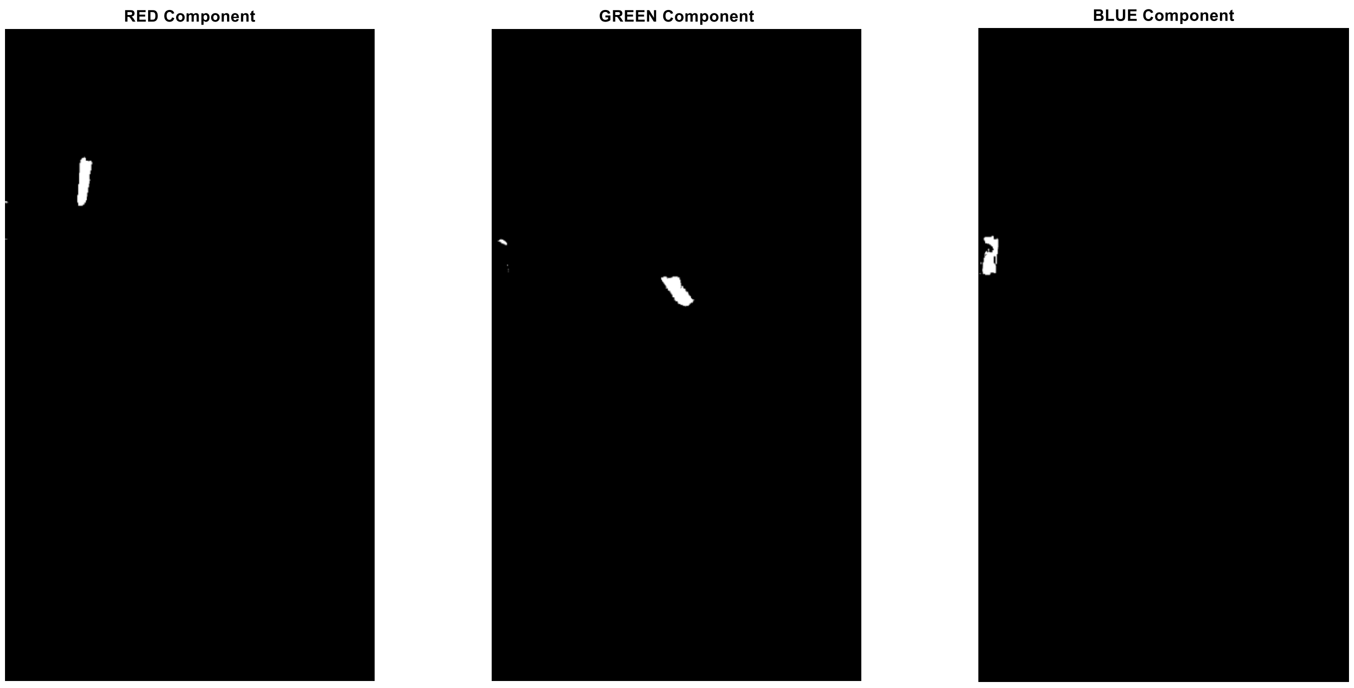
```
    %%Difference Frame to extract red component, DF_CF = Difference
Frame Current Frame
    DF_CF_R=zeros(Size(1),Size(2));
    DF_CF_G=zeros(Size(1),Size(2));
    DF_CF_B=zeros(Size(1),Size(2));
    DF_CF_R = CurrentFrame(:,:,1)-HE_CF;
    DF_CF_G = CurrentFrame(:,:,2)-HE_CF;
    DF_CF_B = CurrentFrame(:,:,3)-HE_CF;
    DF_CF_R = uint8(DF_CF_R);
    DF_CF_G = uint8(DF_CF_G);
    DF_CF_B = uint8(DF_CF_B);
    %%FOR DEBUGGING ONLY
    subplot(1,3,1),    imshow(DF_CF_R), title('RED Component');
    subplot(1,3,2),    imshow(DF_CF_G), title('GREEN Component');
    subplot(1,3,3),    imshow(DF_CF_B), title('BLUE Component');
```



Segmentation/Binarization

The next step is to start separating our object from the background using segmentation by thresholding, in this process the thresholding for the segmentation process was determined by trial and error by analyzing the output image for unwanted noise or having a threshold too high that it does not detect our object.

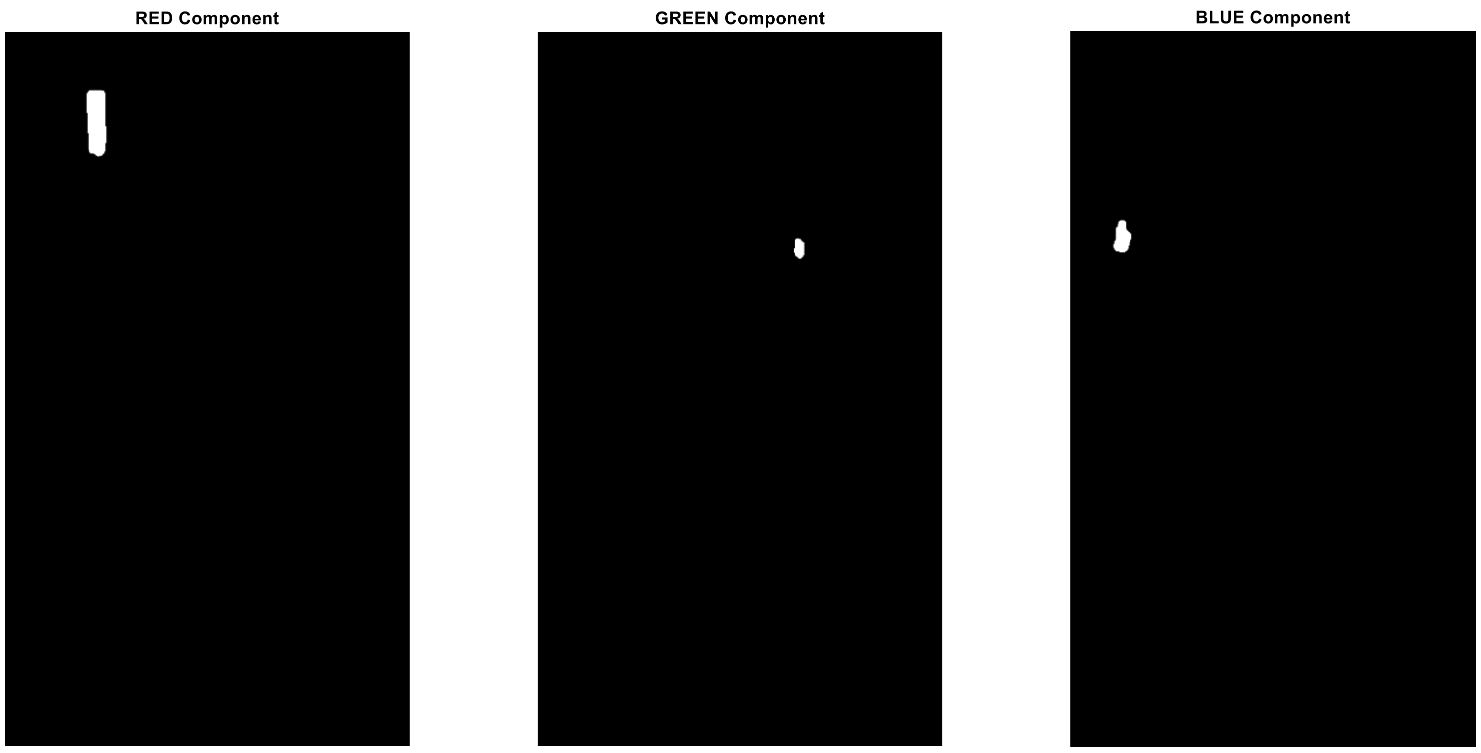
```
%%Binarization using thresholding, f=50, BI_CF = Binarized
Current Frame
    BI_CF_R=zeros(Size(1),Size(2));
    BI_CF_G=zeros(Size(1),Size(2));
    BI_CF_B=zeros(Size(1),Size(2));
    BI_CF_R=uint8(HE_CF);
    BI_CF_G=uint8(HE_CF);
    BI_CF_B=uint8(HE_CF);
    for row = 1:Size(1)
        for column=1:Size(2)
            %%Red frame Binarization
            if DF_CF_R(row,column)> 68
                BI_CF_R(row,column)= 256;
            else
                BI_CF_R(row,column)= 0;
            end
            %%Green Frame BinarizatioN
            if DF_CF_G(row,column)> 58
                BI_CF_G(row,column)= 256;
            else
                BI_CF_G(row,column)= 0;
            end
            %%Blue Frame Binarization
            if DF_CF_B(row,column)> 67
                BI_CF_B(row,column)= 256;
            else
                BI_CF_B(row,column)= 0;
            end
        end
    end
```



Median Filter

Analyzing the segmentation process result we can observe a few occurrences of salt & paper noise, the appropriate filter for this type of noise would be the median filter. The mask size was determined for each color layer by trial and error since some colors had more noise than others due to being similarly colored with background objects, for example, green.

```
BI_CF_R=medfilt2(BI_CF_R,[3 3]);
BI_CF_G=medfilt2(BI_CF_G,[7 7]);
BI_CF_B=medfilt2(BI_CF_B,[3 3]);
subplot(1,3,1),    imshow(BI_CF_R), title('RED Component');
subplot(1,3,2),    imshow(BI_CF_G), title('GREEN Component');
subplot(1,3,3),    imshow(BI_CF_B), title('BLUE Component');
```



Blob Analysis

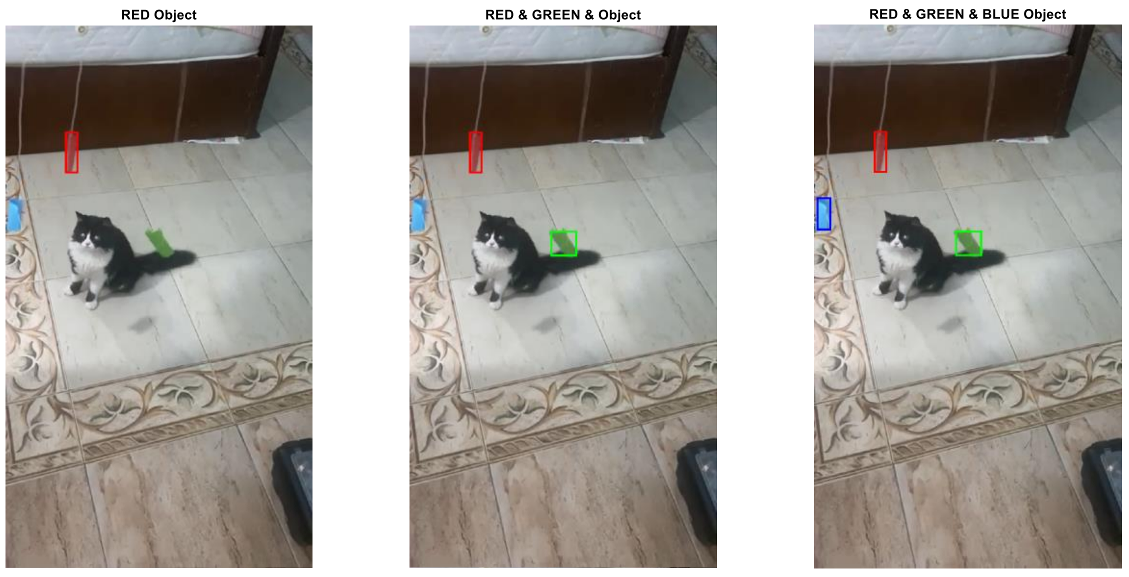
Many methods represent an object by a predefined shape around its centroid or a set of points. Object shape can be defined as a rectangle, ellipse, etc. on the imaging plane or on the physical ground plane. Often such shapes are normalized into a fixed size to simplify feature extraction. In this process we'll use the blob analysis feature of the MATLAB Vision library to determine the Area, Centroid, and Bounding Box coordinates of the objects.

```
%%Initizalizing Blob Objects for each color

blob_R = vision.BlobAnalysis('BoundingBoxOutputPort',
true,'MinimumBlobAreaSource', 'Property','MinimumBlobArea', 150);

blob_G = vision.BlobAnalysis('BoundingBoxOutputPort',
true,'MinimumBlobAreaSource', 'Property','MinimumBlobArea', 150);

blob_B = vision.BlobAnalysis('BoundingBoxOutputPort',
true,'MinimumBlobAreaSource', 'Property','MinimumBlobArea', 150);

      [area_R,centroid_R, bbox_R] = step(blob_R, logical(BI_CF_R));
      [area_G,centroid_G, bbox_G] = step(blob_G, logical(BI_CF_G));
      [area_B,centroid_B, bbox_B] = step(blob_B, logical(BI_CF_B));
    %%Converting BBox into uint16

    bbox_R=uint16(bbox_R);
    bbox_G=uint16(bbox_G);
    bbox_B=uint16(bbox_B);
```

Object Labeling

Having acquired the bounding boxes of each Blob or Object we can start using the Insertshape() function to draw highlighting boxes with the corresponding colors for each object.

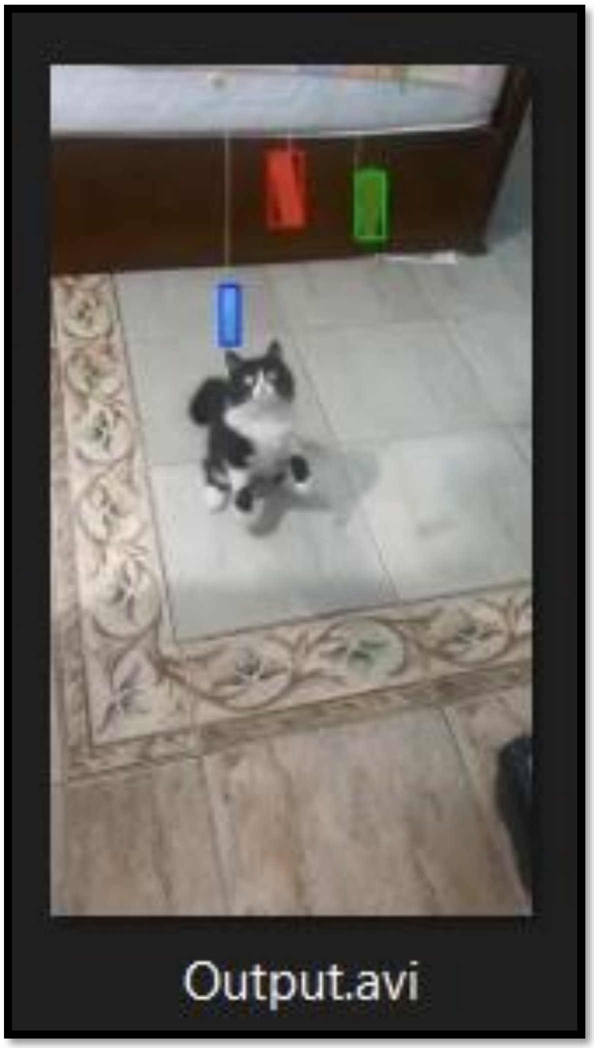
```
vidout=insertShape(CurrentFrame,'rectangle',bbox_R,'LineWidth',3,'Color','red');

vidout=insertShape(vidout,'rectangle',bbox_G,'LineWidth',3,'Color','green');

vidout=insertShape(vidout,'rectangle',bbox_B,'LineWidth',3,'Color','blue');
```



Video Extraction

Once each frame is processed it is added into the open object "V" which is used to stack the frames to output all of them in a video format as an uncompressed AVI.

```
writeVideo(v,vidout)
close(v)
```



References

1. B. B. V. L. Deepak, Real-Time Object Detection and Tracking Using Color Feature and Motion,author profiles for this publication at: https://www.researchgate.net/publication/280111991

2. Baranitharan, A., & Analysis), R. (2021). Real-Time Object Tracking Using MATLAB (Blob Analysis). Retrieved 23 May 2021, from https://www.hackster.io/avinash15101994/real-timeobject-tracking-using-matlab-blob-analysis-358ee8

3. (2021). Retrieved 23 May 2021, from https://www.ripublication.com/irph/ijict_spl/ijictv4n15spl_10.pdf