Object Detection and Tracking using camera

Sandeep Satone
University of Texas at Arlington
701 S Nedderman
sandeep.satone@mavs.uta.edu

Abstract

A video camera is among the most commonly used sensors in a large number of applications, ranging from surveillance to smart rooms for video conferencing. There is a need to develop programs for tasks such as detection, tracking, and recognition of objects, specifically using security cameras The nature of imaging sensors provides ample challenges for detecting objects from the visual sensors. The main focus of this project is to towards highlighting the efficient use of the camera sensors to detect and track objects. We will also discuss the challenges faced during this project and future work that we need to do to improve it. [4] [2] [3] [1]

1. Introduction

Video cameras are fast becoming ubiquitous for a wide range of applications including surveillance, smart video conferencing, markerless human motion capture, animation transfer, video editing and even some critical task like robotics and medical surgery. But there are still challenges in developing camera-based applications. The single video camera is being studied for more than a decade and algorithms are developed to use the functions of a camera to achieve all the above mention tasks.

1.1. What is Object Detection ?

Object Detection is a task of scanning and searching for an object in an image or a video. Video is nothing but a sequence of images with different timestamps. If we stack the images one over another and move them in the sequence we will achieve the illusion of motion. This principle is generally used in the projector to project images on the screen.

1.2. What is Object Tracking ?

In the previous subsection, we have defined Object detection. Now, what is object tracking? Object tracking can be defined as following the motion of the object across the frames of a video. We use object tracking for checking the direction in which the object is moving. We may use it for tracking the speed of the moving object across various image frames or video.

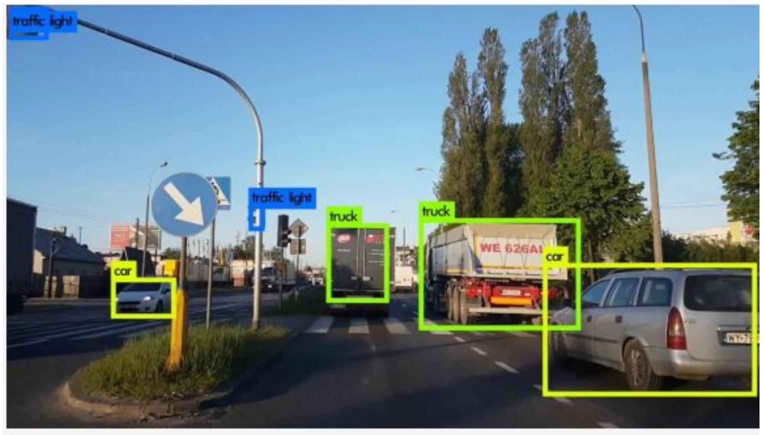


Figure 1. Object Detection Example

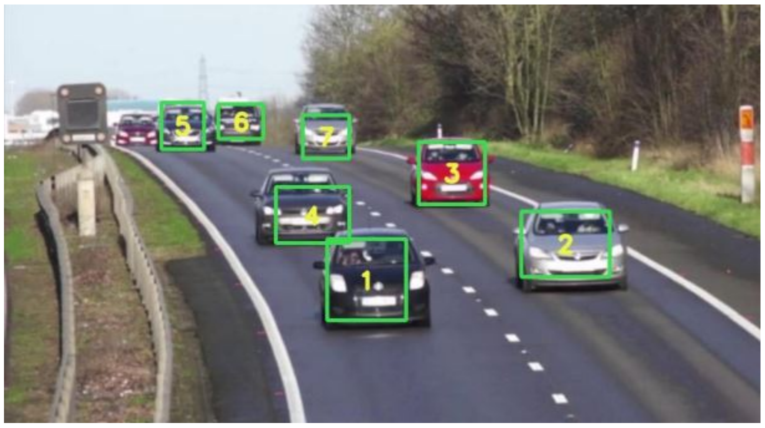


Figure 2. Object Tracking Example

2. Related Work

Generally, there are three different types of approach used in object detection. They are the feature-based approach, motion-based approach, and classifier-based approach. In feature-based approach, the features in the image are extracted as a representation of the image. Detection using colors and shapes are examples of feature-based detection.

In the motion-based approach, the objects are detected based on movements. Hence, only moving objects can be detected while stationary objects cannot be detected. Detection using background subtraction is an example of a motion based approach.

In the classifier-based approach, the classifier is trained to recognize or detect objects by feeding in positive and negative training samples. Example of a classifier-based approach is support vector machine (SVM) classifiers and the cascade of boosted classifiers.

I am using the combination of Feature based and motion based approach to solve this object detection and tracking problem.

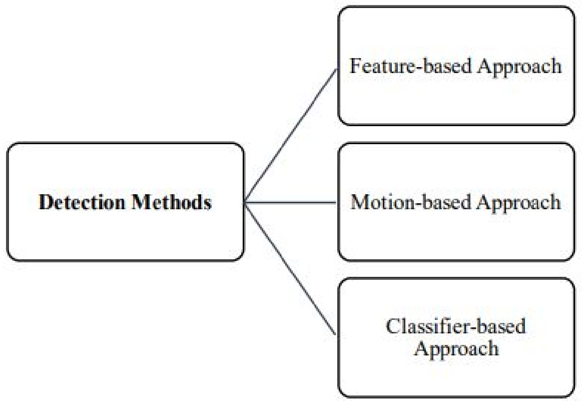


Figure 3. Classification of detection methods

3. Problem Statement

The development of a reliable software in Matlab using existing algorithms to perform object detection and tracking. The computer will be used to model and imitate the human eyes, there are many challenges involved. One of the challenges faced in object detection and tracking is the resolution of the camera and the effect of surrounding light across the object.

All the objects come in various sizes and different colors and shapes. Different shades of various colors like for instance red and pink is easy for humans to detect but it may be difficult to be detected by a computer camera because of the input resolution used by the in-build camera of the computer. The light reflected by various object also affect the resolution based on the light sensitivity of camera.

The other challenge faced during the process what the distance of the object from the camera module. If the object is smaller and is at a long distance from the camera it becomes difficult to identify the object and its color. It becomes nearly impossible to get the object color and tracking once it is beyond few feet.

Therefore it is not easy for one to develop a reliable real-time object detection and tracking system to address the above issues. Although many other researchers have employed different approaches to address these problems. There is still work need to be done to address them all successfully.

4. Problem Solution

Generally, the solution can be divided into two phases. The first phase is the detection phase. After the detection phase, is the tracking phase. The input is taken from the camera and few frames are sampled from the video input of the camera. Around 500 frames are taken to process the object detection and tracking. Each frame is 5 ms apart with a resolution of 1920x1080. The algorithm is implemented with the software tool called Matlab and the camera toolbox and image detection toolbox plugins to execute the program successfully.

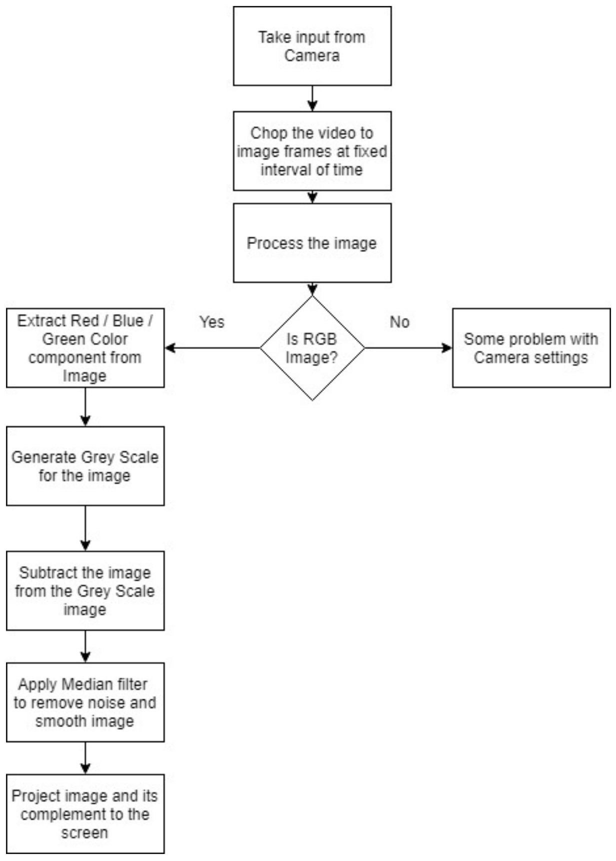

Take input from Camera
Chop the video to image frames at fixed interval of time
Process the image
Is RGB Image?
Yes
No
Extract Red / Blue / Green Color component from Image
Some problem with Camera settings
Generate Grey Scale for the image
Subtract the image from the Grey Scale image
Apply Median filter to remove noise and smooth image
Project image and its complement to the screen

Figure 4. Solution Overview

4.1. Detection Phase

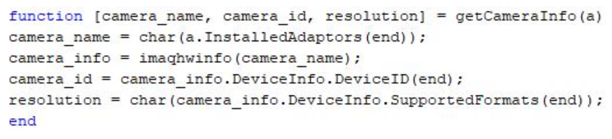
In the beginning, the video is taken as an input from the camera system of the laptop / PC on a frame-by-frame basis. In the feature extraction process, we compute the red/blue/ green color component from the image. Once the color components are extracted the same image frame is then used to generate a grey scale version image. The color component detected is subtracted from the grey scale image. For better result, we apply the Median Filter to remove noise and smooth image. The subtracted image is then converted to binary image to project on the screen. The binary image and the original image are both projected side by side for comparison.

4.2. Tracking Phase

The tracking phase is nothing but the projection of all the detected images one after the other in sequence. This gives the effect of tracking and the user feels as if the camera is tracking it. We also draw a box over the object being tracked and update the coordinates of the tracking box to enclose the object being tracked.

5. Code Explanation

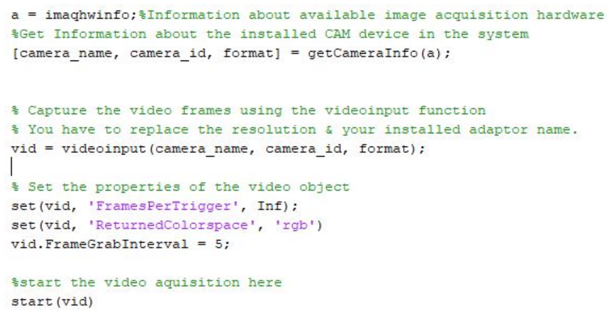
The code written is displayed and explained below.



```
function [camera_name, camera_id, resolution] = getCameraInfo(a)
camera_name = char(a.InstalledAdaptors(end));
camera_info = imaqhwinfo(camera_name);
camera_id = camera_info.DeviceInfo.DeviceID(end);
resolution = char(camera_info.DeviceInfo.SupportedFormats(end));
end
```

Figure 5. Taking camera input

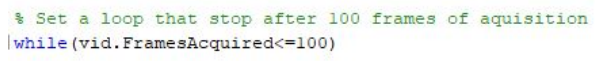
The camera input is taken from the systems in build camera. If the camera is not working this function will display the error message in the console. If the system camera is detected then the camera starts taking input from the camera and generate video of the input.



```
a = imaqhwinfo;%Information about available image acquisition hardware
%Get Information about the installed CAM device in the system
[camera_name, camera_id, format] = getCameraInfo(a);


% Capture the video frames using the videoinput function
% You have to replace the resolution & your installed adaptor name.
vid = videoinput(camera_name, camera_id, format);

% Set the properties of the video object
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb')
vid.FrameGrabInterval = 5;

%start the video aquisition here
start(vid)
```

Figure 6. Process camera input to produce frames of 5ms each

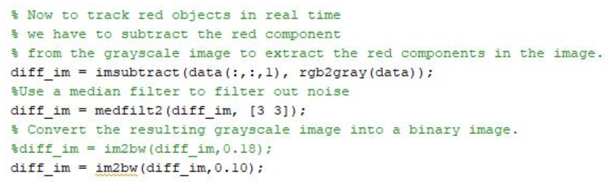
The code above takes the input parameter processed by the camera function and chops the video into frames. For simplicity we are chopping the video to 100 frames with the interval of 5ms each.



```
% Set a loop that stop after 100 frames of aquisition
while(vid.FramesAcquired<=100)
```

Figure 7. Setting loop to process 100 frames

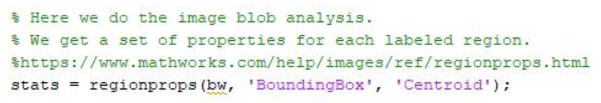
Once the frames are chopped from the video we will process each frame individually. We create a loop to process and track the object in each frame individually. We also project the image once the detection part is done.



```
% Now to track red objects in real time
% we have to subtract the red component
% from the grayscale image to extract the red components in the image.
diff_im = imsubtract(data(:,:,1), rgb2gray(data));
%Use a median filter to filter out noise
diff_im = medfilt2(diff_im, [3 3]);
% Convert the resulting grayscale image into a binary image.
%diff_im = im2bw(diff_im,0.18);
diff_im = im2bw(diff_im,0.10);
```

Figure 8. Extracting features / filtering

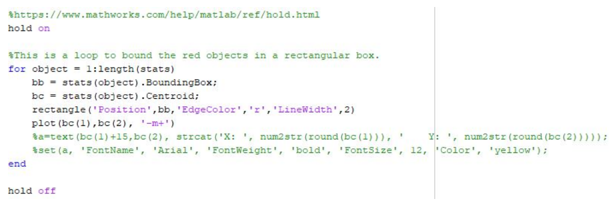
Here we are separating the color component from the image. once the color component is extracted we take the difference of the image from the grey scale image and apply a median filter to it. Median filter smooths the image. Also if the camera resolution is not good. It is recommended to use it. There are many filters for instance median filter, Gaussian filters, etc. I found the median filter gave me the best result for smoothing the image.



```
% Here we do the image blob analysis.
% We get a set of properties for each labeled region.
%https://www.mathworks.com/help/images/ref/regionprops.html
stats = regionprops(bw, 'BoundingBox', 'Centroid');
```

Figure 9. Call Bounding box function for each frame

Once the object is detected in the frame we plot a box around the object. We update the coordinate of the bounding box at each frame.

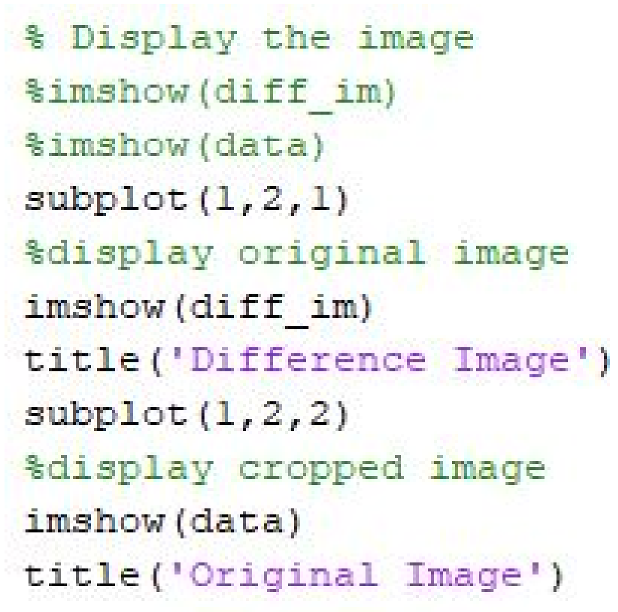


```
%https://www.mathworks.com/help/matlab/ref/hold.html
hold on

%This is a loop to bound the red objects in a rectangular box.
for object = 1:length(stats)
    bb = stats(object).BoundingBox;
    bc = stats(object).Centroid;
    rectangle('Position',bb,'EdgeColor','r','LineWidth',2)
    plot(bc(1),bc(2), '-m+')
    %a=text(bc(1)+15,bc(2), strcat('X: ', num2str(round(bc(1))), '    Y: ', num2str(round(bc(2)))));
    %set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');
end

hold off
```

Figure 10. Creating Bounding box

This function draws the box on the object.

Project the initial image and the difference image side by side. So that the user can see what object is actually being detected. The left side shows the binary image of the difference which is basically black and white image of the original image. The detected object is displayed as empty pixels (The white part of the image). To the right we display the original image.



```
% Display the image
%imshow(diff_im)
%imshow(data)
subplot(1,2,1)
%display original image
imshow(diff_im)
title('Difference Image')
subplot(1,2,2)
%display cropped image
imshow(data)
title('Original Image')
```

Figure 11. Displaying Image on screen

6. Experimental Results

The below images displays the experimental output for the project.

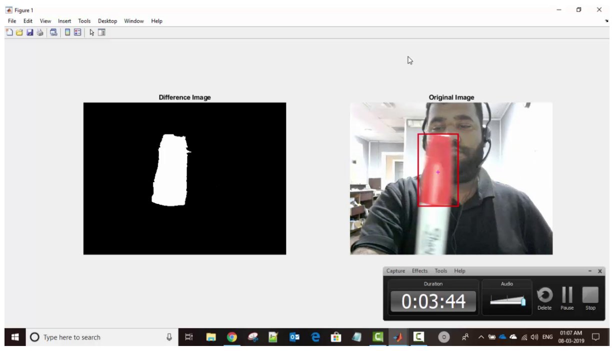


Figure 12. Red Object detection

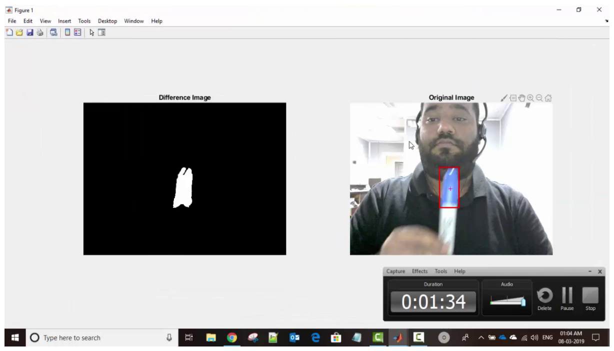


Figure 13. Blue Object detection

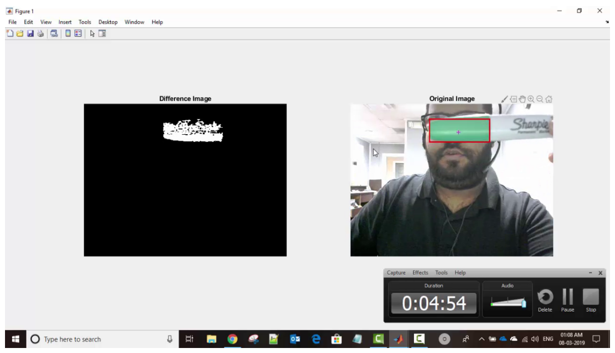


Figure 14. Green Object detection

7. Project Code

```
// getCameraInfo.m
function [camera_name, camera_id,
    resolution] = getCameraInfo(a)
camera_name =
    char(a.InstalledAdaptors(end));
camera_info = imaqhwinfo(camera_name);
camera_id =
    camera_info.DeviceInfo.DeviceID(end);
resolution =
    char(camera_info.DeviceInfo.SupportedFormats(end));
end
```

```
// Main.m

%https://www.mathworks.com/help/imaq/imaqhwinfo.html
a = imaqhwinfo;%Information about
    available image acquisition hardware
%Get Information about the installed CAM
    device in the system
[camera_name, camera_id, format] =
    getCameraInfo(a);


% Capture the video frames using the
    videoinput function
% You have to replace the resolution &
    your installed adaptor name.
vid = videoinput(camera_name, camera_id,
    format);

% Set the properties of the video object
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb')
vid.FrameGrabInterval = 5;

%start the video aquisition here
start(vid)

% Set a loop that stop after 100 frames of
```

```
    aquisition
while(vid.FramesAcquired<=100)

   % Get the snapshot of the current frame

   data = getsnapshot(vid);%immediately
       returns one single image frame

   % Now to track red objects in real time
   % we have to subtract the red component
   % from the grayscale image to extract
       the red components in the image.
   diff_im = imsubtract(data(:,:,1),
       rgb2gray(data));
   %Use a median filter to filter out noise
   diff_im = medfilt2(diff_im, [3 3]);
   % Convert the resulting grayscale image
       into a binary image.
   %diff_im = im2bw(diff_im,0.18);
   diff_im = im2bw(diff_im,0.10);

   % Remove all those pixels less than
       300px

   diff_im = bwareaopen(diff_im,100);

   % Label all the connected components in
       the image.

   bw = bwlabel(diff_im, 8);

   % Here we do the image blob analysis.
   % We get a set of properties for each
       labeled region.

   stats = regionprops(bw, 'BoundingBox',
       'Centroid');

   % Display the image
   %imshow(diff_im)
   %imshow(data)
   subplot(1,2,1)
   %display original image
   imshow(diff_im)
   title('Difference Image')
   subplot(1,2,2)
   %display cropped image
   imshow(data)
   title('Original Image')


   hold on

   %This is a loop to bound the red
       objects in a rectangular box.
   for object = 1:length(stats)
      bb = stats(object).BoundingBox;
      bc = stats(object).Centroid;
      rectangle('Position',bb,'EdgeColor','r','LineWidth'
      plot(bc(1),bc(2), '-m+')
      %a=text(bc(1)+15,bc(2), strcat('X:
          ', num2str(round(bc(1))), ' Y: ',
          num2str(round(bc(2)))));
      %set(a, 'FontName', 'Arial',
          'FontWeight', 'bold', 'FontSize',
          12, 'Color', 'yellow');
   end

   hold off
end
% Both the loops end here.

% Stop the video aquisition.
stop(vid);

% Flush all the image data stored in the
    memory buffer.
flushdata(vid);

% Clear all variables
clear all
sprintf('%s','Project demonstration end
    here !!! ')
```

8. Conclusion

At the end of this project, we have developed an Object detection and tracking system. We have shown that the regular camera of a computer can be used for this task. We have also learned that using a good experimental setup and good camera we can improve the results. We used the Matlab system to process our data but we can also use other technology like python. Python has some specific libraries which can be used to develop the algorithm for object detection.

9. Future Work

Although we have done the experiment successfully. We can improve and customize this in the future. The complete project is dependent on the resolution of the camera. If we use a bad resolution camera we can't get accurate results. We need to work upon the ways we can reduce the error in color detection due to poor camera resolution.

Also, the object is not detected properly in improper lighting. We need to implement some logic to improve the quality of the frame generated for the low light experimental setup.

In the future, we also need to improve upon object detection algorithm, for distant objects. The distant objects cannot be properly detected with a low-resolution camera. The camera sometimes also does not recognize the difference between slightly different shades of the color if the distance of the object is more than the range of the camera.

The above project can also be implemented with the Convolutional Neural Networks. But the setup for the experiment will be slightly different.

10. references

References

[1] t. Multi-object tracking using color and motion.
[2] I. to the Special Section on Video Surveillance.
[3] C. B. P. Tracking.
[4] P. tracking by detection and people detection by tracking.