Efficient Color Based Object Detection and Tracking

Ahsan Mustafa
Electrical and Electronics Engineering
Middle Eastern Technical University
Abbottabad, Pakistan
ahsanmustafa30@gmail.com

Abstract— In this paper, a new method to detect and track objects using color is discussed, which is very efficient compared to the regular methods. It is based on a new formula proposed by an author to convert RGB image into an Intensity image. This formula has a distinct feature of highlighting any primary color of interest and suppressing the rest. This discriminating ability is used to detect the object of any primary shade efficiently. This unique feature eliminates many of the standard steps, such as segmentation and histogram matching, normally used in color-based tracking models. This paper discusses about the modification of the original formula so that it can be applicable for other colors rather than just for primary.

I. INTRODUCTION

Using color for detecting and tracking moving objects in videos is a major research area of Machine vision. The main goal is to simulate the basic abilities of biological systems, whether it is better recognition of surroundings or analysis of a crowd to detect vehicles and people. Finding an object of interest in a scenario is known as object detection whereas object tracking means to find the trajectory of the moving object in a frame. For instance, if surveilling a road is being done, and a moving car needs to be detected would be referred as object detection and to estimate the direction and the path that the car has taken in time, another frame would be taken to track its trajectory, is known as object tracking.

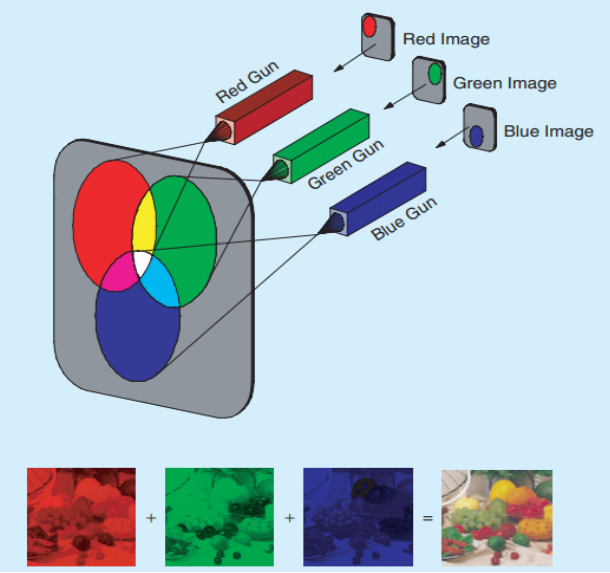
There are many ways and techniques to detect and track an object in videos. Most used method is known as feature-based method, in which certain features such as gradient, color etc. are extracted from frames to detect and track objects. Other methods such as background subtraction, optical flow etc. are also used to extract different features for the purpose of detection and tracking the object of interest. Of all the features, one of the major ones used in this process is color, which is very helpful in object recognition.

This paper is based on the formula proposed by Rachna Verma, which is efficient for detection and tracking of objects with primary color shades, and this paper discusses the modified formula which works for other colors than the primary. This proposed methodology takes a RGB image and converts it to an intensity image using this formula which suppresses the other colors as well as highlights the color of interest. This provides us with a big bonus in reducing the computation since it does not use background modelling, foreground subtraction, histogram matching etc. hence making it a lot more efficient than the regular models used for this purpose.

This paper will have 5 further sections, where firstly the background and different color spaces will be discussed to give a general overview on the topic. Secondly, how those basic color-based concepts can be utilized in different implementation, where the focus will be on detection and tracking the object of interest. Thirdly, the proposed system for this method will be explained. Fourthly, the experimental results using the new proposed formula for combination of primary colors will be presented. Finally, the last section will conclude this paper and discuss about its various implementations as well as how to improve it in the future.

II. COLOR SPACES ADDITIVE SYSTEM AND IMAGE FORMATION

To understand, the proposed formula and how it distinguishes amongst color highlighting one and suppressing the rest, color spaces such as RGB, BGR, CMYK, HSV etc. are a good place to start with. Using combination of primary colors, red, green, and blue, in Machine Vision and computer systems colors gives us other colors such as yellow, cyan, magenta, and white. This is regarding RGB color space whereas other color spaces comprise of other primary colors and their combinations result in different colors. A colored image comprises of three dimensions, in RGB case i.e., Red, Green and Blue, where combining the three dimensions result in a colored image, seeable on the screen as shown in Fig. 1 below [1].



.

Fig. 1. Additive RGB Color System

CMYK can be said as an inverse of RGB color system which uses cyan, magenta, and yellow as its primary colors, where their different combinations result in red, blue, and green. A colored image can be seen in various color spaces,

in this paper the focus will be on RGB. An image has different intensity for each color which affects the look of it. This is shown in Fig. 2 below, this can be used to an advantage by using the proposed formula i.e., highlighting the color of interest while suppressing the rest.

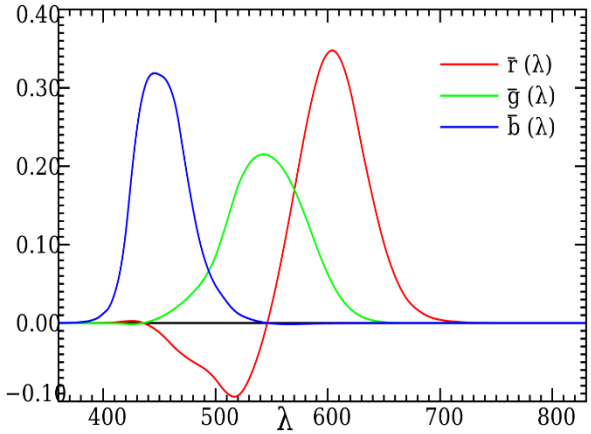


Fig. 2. Intensities of an image in RGB Color System

. Fig. 2 shows intensity values of an image in respect to RGB color representation scheme. As seen, each pixel has its own (R, G, B) values. This is a key information to exploit for detection of an object. For instance, a pixel value with (18, 120, 7) would look much greener compared to a pixel value of (160,200, 180) as it has a greater difference compared to red and blue, even though the green pixel value in the second pixel is greater. These ratios amongst R, G, B values will be used to extract the color of interest and suppressing the others with the help of the formula further explained in section IV.

III. COLOR BASED OBJECT DETECTION AND TRACKING METHODOLOGY

Flowchart shown in Fig. 3 below depicts the how an object is detected and tracked, by using color features of an image, which is one of the most used features of an image to perform detection and tracking. There are many challenges faced during tracking an object such as illumination, change in object orientation, and inaccuracy of the color filters, etc. as the main concept of tracking is to map the detected object, to compute its trajectory in two consecutive frames of a video. To track object, firstly detection needs to be done, which generally is performed in several steps, increasing computation inefficiency. To make it more effective regarding computation efficiency, the proposed formula is used which is explained in section IV with detail.

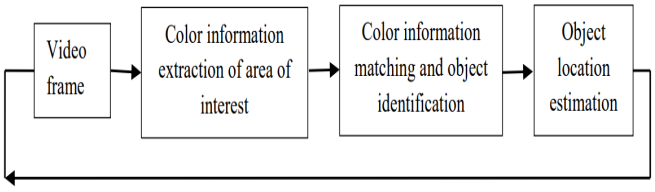


Fig. 3. Flowchart for Color-based Object Detection and Tracking

General methodology to detect objects is by modelling objects using color histograms, as they are cheap computationally and invariant to scale and rotation. To find the location of the objects Mean-Shift Algorithm is used, which finds an image window with most similarity to the object's color histogram in the current frame. Afterwards, for computing similarity amongst the two histograms Manhattan, Euclidean, or Bhattacharya Distance etc. can be used. The challenge locating method faces is that it has problems to manage cluttering and scale variations. There are other implementations to further improve this process which usually have tradeoffs, improving one aspect while compromising on another.

In this paper, the proposed method, by Rachna Verma, for new intensity calculation strategy has been implemented such that it aids in the autonomous detection of primary as well as their respective combinational colors more efficiently. The formula and its modifications to extract further information from the image are discussed in detail below.

IV. PROPOSED METHOD AND INTENSITY FORMULAS

Highlight of this paper is the formula for a very effective conversion of an RGB frame to an intensity frame. This formula is found to be very practical because of its discriminating ability in highlighting primary as well as secondary colors in an image, where the generated intensity image is further processed by thresholding it with a predefined value to further clarify the segmented object. The proposed formula for different colors, firstly for the primary and then for secondary colors will be shown below, as well as the resulting intensity image after thresholding the image with a predefined value. The proposed formula can be configured in respect to the primary color wanted to be detected in a video frame. For instance, for red, it would be as follows in equation i [2].

$$I = \frac{R * R}{G * B} \qquad (i)$$

Where I is the calculated intensity of the pixel while R, G, B are the pixel values. So, to highlight green and blue, automatically suppressing the rest, the formula for calculation of Intensity can be modified as follows, shown in equation ii and iii.

$$I = \frac{G * G}{R * B} \qquad (ii)$$

$$I = \frac{B * B}{R * G} \qquad (iii)$$

To accommodate secondary colors, to have an effective intensity image as a result, some altercation of this formula is required. To do this, combinations of these primary colors in different manners to generate new parameters is required. As [Fig1] shows that red, green, and blue, can be combined to give yellow, cyan, and magenta. Further modification of this formula and thresholding it with a predefined value aids to accomplish detection of the desired color, whether it is yellow, cyan, or magenta. Along with primary color segmentation, secondary color segmentation will also be shown below. Before moving onto the figures, the new parameters and the aforementioned formula's modification will be discussed. For instance, for yellow, combination of

red and green, formula can be seen in equation iv. The new parameter introduced to this formula is regarding the suppression of red and green, while highlighting their common regions i.e., yellow. This is achieved by using the logical operator "OR" between red and green pixel value.

L = R "OR" G

$$I = \frac{R * G}{L * B} \qquad (iv)$$

Where L is the result of logical OR operation between red and green and I is the calculated intensity pixel value for yellow color. Similarly, for Magenta and Cyan, two new parameters and the modified formula for each are as follows.

K = R "OR" B

$$I = \frac{R * B}{K * G} \qquad (v)$$

J = G "OR" B

$$I = \frac{G * B}{J * R} \qquad (vi)$$

Fig. 4 shows the resultant output images after applying the respective formulas mentioned above to segment the color of interest. Red, in bright lighting conditions, followed by green in dark conditions and finally yellow, in different scenarios, resultant intensity images are shown below. The resulting intensity images have also been threshold with a predefined value to remove the noise in the image.

After confirming that detection of the desired object in each video frame is successful, using the proposed formula, tracking of the detected object which is moving in a video becomes the focus. Fig. 5 shows a block diagram of a proposed system which combines detection and tracking of the object. After detection of the desired object, computation of centroid of the segmented image is done. To remove the noise from the image median filtering of the frame is done and after conversion to an intensity image, it is further processed by thresholding with a predefined value. The resultant is used to compute the centroid with the help of moments and this process is done again for the next frame. Simply by seeing the trajectory of the centroid over several frames, tracking of the object is done successfully. The workflow and its implementation will be shown with the help of experimental results in SECTION V.

V. PROTOTYPE SYSTEM AND EXPERIMENTAL RESULTS

The prototype system is shown in Fig. 5 which will be followed by experimental results, showing the locations of centroid of moving object on each frame of the video. This system has been developed to test the concepts explained above, as well as to check its endurance under different illuminations. Fig. 6 will show the computed centroid on intensity images whereas Fig. 7 and Fig. 8 will show the

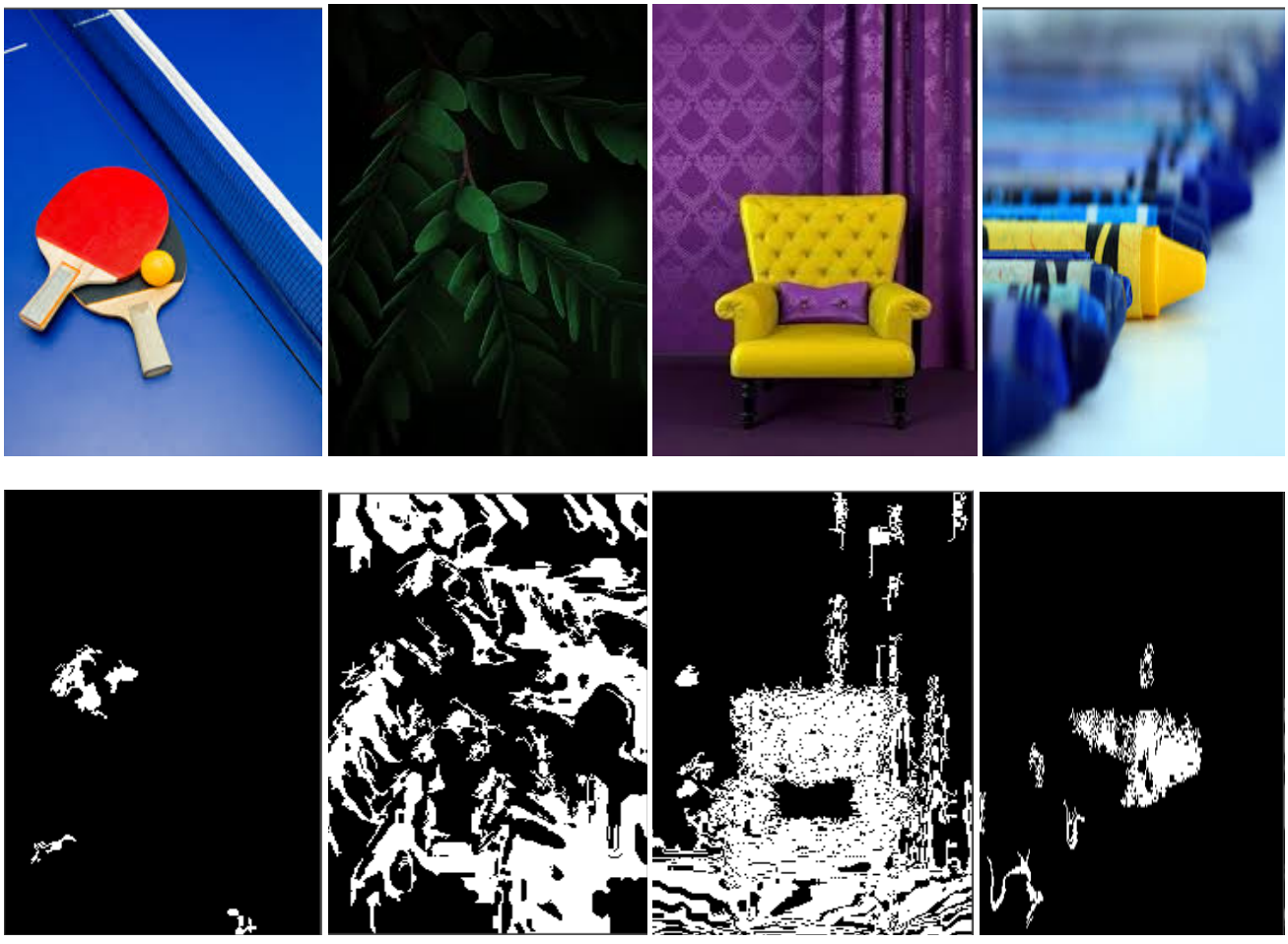


Fig. 4 Sample images under different illuminations and their intensity images

implementation and the computed trajectory of the object we want to detect. First implementation would be for a primary color while the second implementation will track a secondary color object.

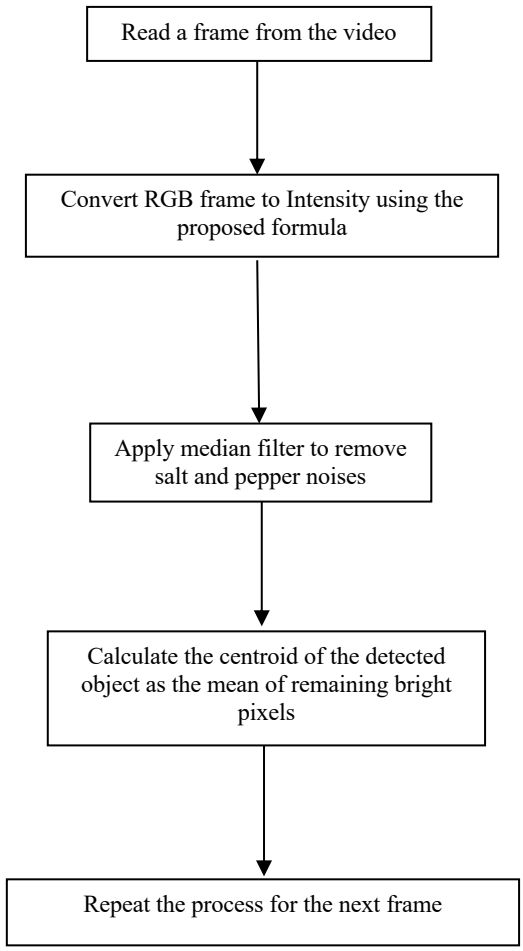


Fig. 5 Block Diagram of tracking object algorithm

The prototype system, shown above in Fig. 5, has been made using a static camera, Core i3 CPU Laptop running on Windows 10. The video has been tested under different lighting conditions to check the algorithm's robustness. Python OpenCV has been used as the programming language.

In Fig. 6 shown below, after processing the intensity image to detect a secondary color i.e yellow, the remaining bright pixels are used to calculate the centroid values.

In Fig. 7 and Fig. 8 red smartphone and a yellow cup are being tracked in a video. Their computed centroids over the frames are shown for confirmation of successful tracking. Both are shown on the next page.

VI. CONCLUSION

Detection and tracking of moving objects is a wide research area with many practical applications. A lot of work has been done in this area for purposes such as human machine interaction, generation of trajectory, surveillance etc. This paper works on the proposed Intensity formula by Rachna Verma and provides with a new model to detect object and track for not only primary colors but every color in RGB color space. With the inclusion of new parameters, the RGB image is converted to an intensity image where a secondary color is highlighted while the rest is suppressed. This image is then further filtered to avail the color of interest in the frame. Traditional use of computationally expensive processes are eliminated with this model such as segmentation, background modelling and histogram matching. This model can be further improved in the future by making it more robust to illumination, occlusions and improving the accuracy in complex color schemes where the object of interest is harder to detect. Previously this formula was limited only to the use of primary colors, this paper's main contribution is that now it can work for secondary colors as well. In future, attempts will be made to extend its applicability under more harsher conditions as well as improving the formula for further accuracy and more variety.

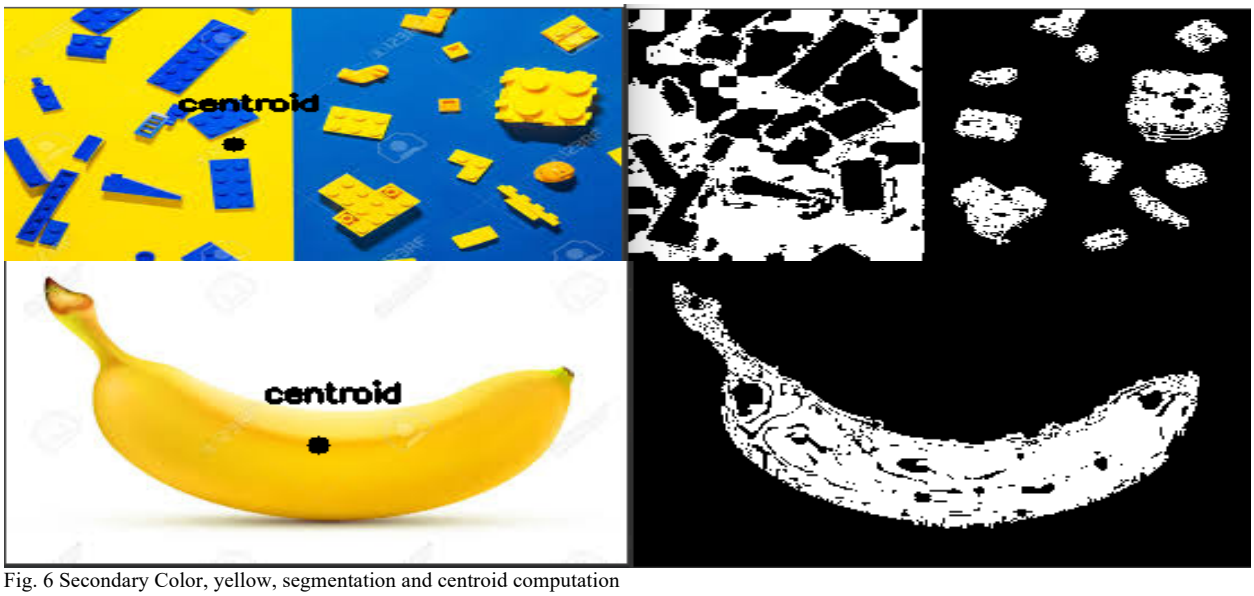


Fig. 6 Secondary Color, yellow, segmentation and centroid computation

Fig. 7 Red color smartphone being tracked, with the black dot shown as the centroid in a video with frames taken every millisecond from video

Fig. 8 Yellow color cup being tracked, with the black dot shown as the centroid in a video with frames taken every millisecond from video

VII. REFERENCES

[1] H. J. Trussel, E. Saber, M. Vrhel, “Color image processing: Basics and special issue overview” IEEE Signal processing magazine, Vol. 22 (No. 1), pp. 14-22, 2005.

[2] R. Verma “An efficient color-based object detection and tracking in videos” International journal of Computer Engineering and applications, Vol. XI, Issue XI, pp. 172-178, November 2017.

VIII. APPENDIX

Attached is my code which I created using Python OpenCV, for project purposes.

```
import cv2
import numpy as np
camera_port = 0
cap = cv2.VideoCapture(camera_port,
cv2.CAP_DSHOW)

cap.set(3,600)
cap.set(4,400)
zeros = None

while True:
    ret_value, frame = cap.read()

    if ret_value == True:
        #adding median blurring to
```

```
smooth the image
        frame = cv2.medianBlur(frame, 5)

        #splitting the image into frames
        b, g, r = cv2.split(frame)
        #time.sleep(1)
        l = np.logical_or(r, g)
        #time.sleep(1)
        ot1 = np.multiply(r, r)
        ot2 = np.multiply(g, b)
        ot1 = ot1.astype(np.float)
        ot2 = ot2.astype(np.float)
        intensity = np.floor_divide(ot1,
ot2)

        ret, thresh1 =
cv2.threshold(intensity, 1, 255,
cv2.THRESH_BINARY)
        image1copy = np.uint8(thresh1)
        # doing connected components
processing
        nlabels, labels, stats,
centroids =
cv2.connectedComponentsWithStats(image1c
opy, None, None, None, 8, cv2.CV_32S)

        # get CC_STAT_AREA component as
stats[label, COLUMN]
        areas = stats[1:,
cv2.CC_STAT_AREA]

        result =
np.zeros((labels.shape), np.uint8)

        for i in range(0, nlabels - 1):
            if areas[i] >= 900:  # keep
                result[labels == i + 1]
= 255
                #contours
                contours0, hierarchy =
cv2.findContours(result.copy(),
cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)
                # calculate moments of
binary image
                M = cv2.moments(result)
                # calculate x,y
coordinate of center
                if M["m00"] != 0:
                    cX = int(M["m10"] /
M["m00"])
                    cY = int(M["m01"] /
M["m00"])
                else:
                    cX, cY = 0, 0
                # put text and highlight
the center
                cv2.circle(frame, (cX,
cY), 5, (0, 0, 0), -1)
                # showing the centroid
on frame
                cv2.imshow("centroid",
frame)
            k = cv2.waitKey(10)
            if k == 27:
                break


cap.release()
cv2.destroyAllWindows()
```