International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 5 Issue 3, March-April 2021 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Color Based Object Tracking with OpenCV - A Survey

Vatsal Bambhania[1], Harshad P Patel[2]

[1]Student, [2]Lecturer,

[1]Computer Engineering Department, L.J. Institute Engineering and Technology, Ahmedabad, Gujarat, India
[2]Instrumentation & Control Engineering Department, Government Polytechnic, Ahmedabad, Gujarat, India

ABSTRACT

Object tracking is a rapidly growing field in machine learning. Object tracking is exactly what name suggests, to keep tracking of an object. This method has all sorts of application in wide range of fields like military, household, traffic cameras, industries, etc. There are certain algorithms for the object tracking but the easiest one is color-based object detection. This is a color-based algorithm for object tracking supported very well in OpenCV library. OpenCV is an library popular among python developers, those who are interested in Computer vision. It is an open source library and hence anyone can use and modify it without any restrictions and licensing. The Color-based method of object tracking is fully supported by OpenCV's vast varieties of functions. There is little bit of simple math and an excellent logic behind this method of object tracking. But in simple language the target object is identified from and image given explicitly by user or some area selected from frame of video, and algorithm continuously search for that object from each frame in video and highlights the best match for every frame. But like every algorithm it also has some pros and cons which are discussed here.

How to cite this paper: Vatsal Bambhania | Harshad P Patel "Color Based Object Tracking with OpenCV - A Survey" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-5 | Issue-3, April 2021, pp.802-806, URL: www.ijtsrd.com/papers/ijtsrd39964.pdf

IJTSRD39964

1. INTRODUCTION

1.1. OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV is built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library comes handy with more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, Video Surf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan. Thus there are tons of uses, as some mentioned above, of Object detection technique.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. A full-featured CUDA and OpenCL interfaces are being actively used. OpenCV is written natively in C++ and has a template interface that works seamlessly with STL containers.

1.2. Object Recognition

Object recognition is the area of artificial intelligence (AI) concerned with the abilities of AI and robots implementations to recognize various things and entities. Object recognition allows AI programs and robots to pick out and identify objects from inputs like video and still camera images. Methods used for object identification include 3D models, component identification, edge detection and analysis of appearances from different angles.

Object recognition is at the convergence points of robotics, machine vision, neural networks and AI. Google and Microsoft are among the companies working in this area, e.g. Google's driverless car and Microsoft's Kinect system both use object recognition. Robots that understand their environments can perform more complex tasks better. Major advances of object recognition stand to revolutionize AI and robotics. MIT has created neural networks, based on our

understanding of how the brain works, that allow software to identify objects almost as quickly as primates do. Gathered visual data from cloud robotics can allow multiple robots to learn tasks associated with object recognition faster. Robots can also reference massive databases of known objects and that knowledge can be shared among all connected robots.

But all of these needed to be started at some point, and that is where the OpenCV comes in play. Thus OpenCV is the tool that is used to accomplish the task of object detection.

2. Mathematics of the Object Detection

2.1. Histograms

Before getting into the mathematics behind the object detection first we need to understand how an image is represented in the computer. An image is nothing but the array of number representing some values, specifically the value of color that is to be displayed for each pixel. The color is represented in any form. It can be RGB, HSV, Gray Scale, etc. A histogram is a graphical display of data using bars of different heights. In it, each bar group is partitioned by specific ranges. Taller bars show that more data falls in that range. A figure-1 below shows histogram. It displays the shape and spread of continuous sample data.

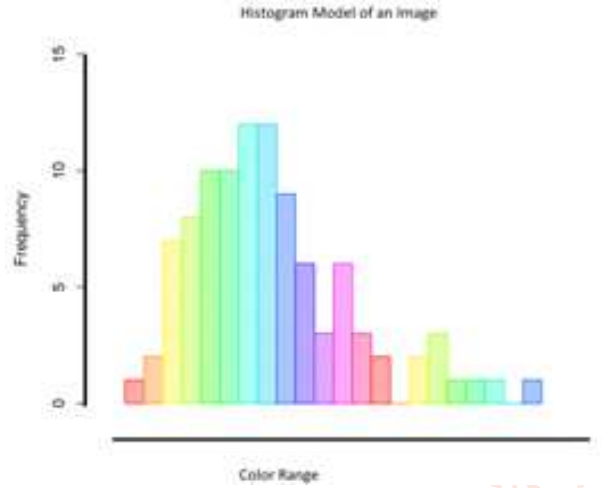


Figure 1 Histogram Model

Histogram or Histogram model is a plot of the frequency against their matching reflectance values. For images the values is the value of pixel color in the image. As said above image is the histogram of in image. Y-axis represents the frequency of particular color and X-axis represents corresponding color. This color range contains all the colors those are in the image. To make the histogram model from the source image the OpenCV contains the inbuilt function cv2.calcHist().

2.2. Back Projection on and Image

2.2.1. What is Back Projection

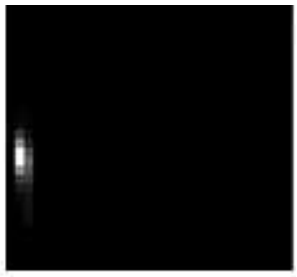
First of all if we want to detect the object from histogram the best approach is to use Back Projection as it quick, requires less computational power and its working can be easily understood. Thus the Back Projection is first step for object tracking.

Back Projection is a way of recording how well the pixels of a given image fit the distribution of pixels in a histogram model. To make it simpler, for Back Projection, we calculate the histogram model of a feature and then use it to find this feature in an image. Application example: If you have a histogram of flash color (say, a Hue-Saturation histogram), then you can use it to find flash color areas in an image.

2.2.2. How Back Projection works

In order to understand the working of Back Projection let's take an example of detecting the skin. Suppose we have skin histogram(Hue-Saturation) as shown in image below in figure-2. The Histogram below is the histogram model of the skin (which we know represents the sample of skin), also we had applied some mask to capture histogram of skin only.

Skin Image

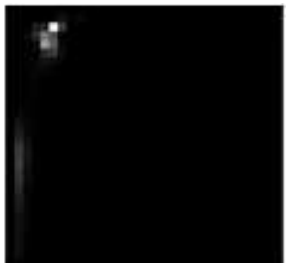


Histogram Model

Figure 2

Now to test other sample of skin like one shown below in figure-3.

Skin Image



Histogram Model

Figure 3

1. In each pixel of our Test Image (i.e. p(i,j)), collect the data and find the correspondent bin location for that pixel (i.e. (h(i,j) , s(i,j))).
2. Lookup the model histogram in the correspondent bin (bar of corresponding color) e.g (h(i,j) , s(i,j)), and read the bin value.
3. Store this bin value in a new image (Back Projection). Also, you may consider to normalize the model histogram first, so the output for the Test Image can be visible for you.
4. Applying the steps above, we get the Back Projection image for our test image as shown in figure-4.

Figure 4 Back Projection Test Image

2.3. MeanShift

A MeanShift is an algorithm, a non-parametric algorithm, use to analyze the feature space to find the maxima of density function. The intuition behind the MeanShift is simple. Consider you have a set of points. (It can be a pixel distribution like histogram Back Projection). You are given a small window (may be a circle) and you have to move that window to the area of maximum pixel density (or maximum number of points). It is illustrated in the simple image given below in figure-6.

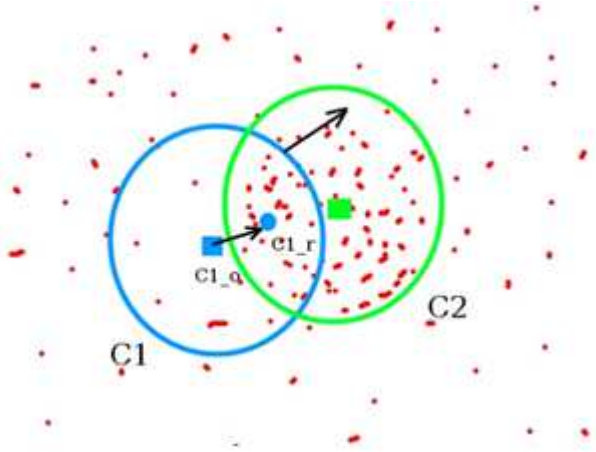


Figure 6 MeanShift Sample Image

The initial window is shown in blue circle with the name "C1". Its original center is marked in blue rectangle, named "C1_o". But if you find the centroid of the points inside that window, you will get the point "C1_r" (marked in small blue circle) which is the real centroid of the window. Surely they don't match. So move your window such that the circle of the new window matches with the previous centroid. Again find the new centroid. Most probably, it won't match. So move it again, and continue the iterations such that the center of window and its centroid falls on the same location (or within a small desired error). So finally what you obtain is a window with maximum pixel distribution. It is marked with a green circle, named "C2". So we normally pass the histogram back projected image and initial target location. The target location is specified manually, hard coded coordinates, or selected from the video frame. When the object moves, obviously the movement is reflected in the histogram back projected image. As a result, the MeanShift algorithm moves our window to the new location with maximum density.

Meanshift using OpenCV library:

```
###
_, frame = cap.read()
hsvFrame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
bp = cv2.calcBackProject([hsvFrame], [0], roi, [0, 180], 1)
_, object_tracker = cv2.meanShift(bp, object_tracker, termn)
x, y, width, height = object_tracker
result = cv2.rectangle(frame, (x, y), (x + width, y + height), 255, 2)
cv2.imshow('MeanShift Demo', result)
###
```

But even with this hectic math there comes a flaw with this method, that is the size of the window is constant hard coded and it won't change with the change in the size of target object.

2.4. CamShift

As mentioned the flaw in MeanShift above, CamShift is similar algorithm to MeanShift but aimed to overcome the flaw of MeanShift. In the algorithm the size of window is initially defined but as video continues the window size also changes in response to change in size of target object.

In first step of CamShift it applied the MeanShift to calculate the density and find displacement of object from it. Once the MeanShift converges, it updates the size of window. It also calculates the orientation of the best fitting ellipse to it. Again it applies the MeanShift with new scaled search window and previous window location. The process continues until the required accuracy is met. Example of CamShift is as shown below in figure-7.

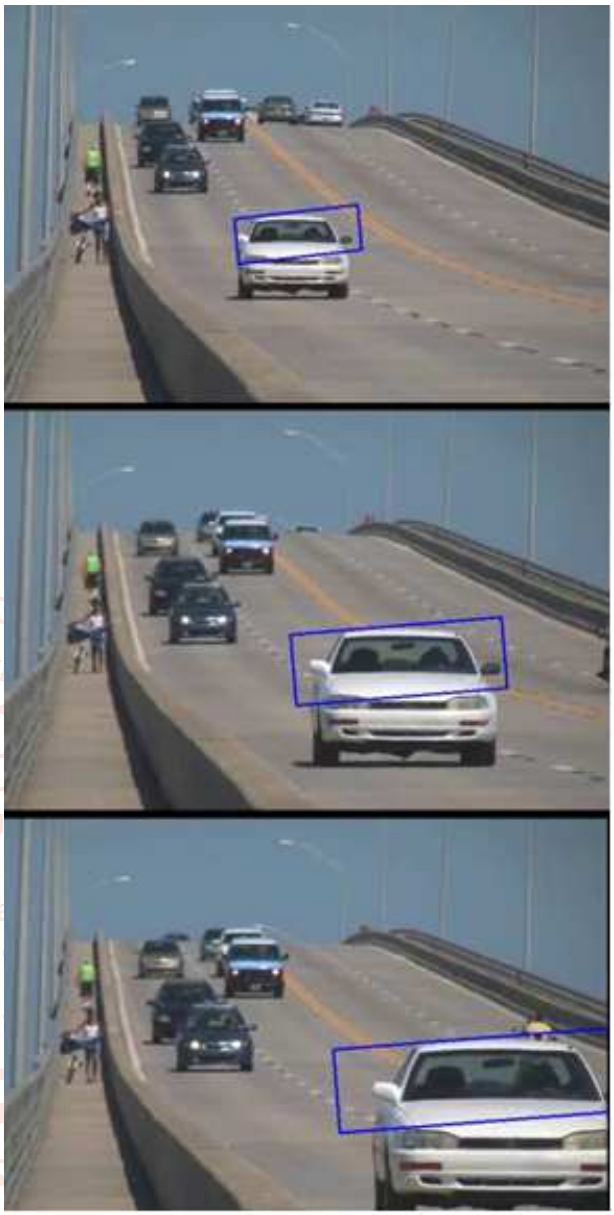


Figure 7 CamShift Sample Image

Continiously Adatpive Meanshift(CAM) using OpenCV library:

```
###
_, frame = cap.read()
hsvFrame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
bp = cv2.calcBackProject([hsvFrame], [0], roi, [0, 180], 1)
_, object_tracker = cv2.CamShift(bp, object_tracker, termn)
x, y, width, height = object_tracker
result = cv2.rectangle(frame, (x, y), (x + width, y + height), 255, 2)
cv2.imshow('CamShift Demo', result)
###
```

About the methods in above snippet:

calcBackProject:

This is a build-in method in opencv that calculates the back projection of our roi to current frame. In other words, this method will calculate how well our roi (region of interest) fits with pixel distribution of image.

inRange:
This method is use to filter image on using a given range of color. Also it a build-in method of openCV library.

3. Applying the about Math to track Object

3.1. How the Object tracking works at Backend

Now as we had seen the math behind the object tracking, now it's time to apply it on video to track an object. The video that will be given as input will be analyses frame by frame. Thus we can control the speed of video by controlling the frames per second (fps).

In order to detect an object following steps has to be performed.

1. First of all the target object is detected from the video
2. Now this detected object is tracked frame by frame as video goes on

First of all we need to detect the target object which we want to keep track of in video, and in order to detect an object we need to have an image of that target object. This image we will use to create the histogram model of target object. After creating the histogram model of target image we feed that model to the Back Projection algorithm. Thus as a result we get a box's starting coordinate in 2-D space of image and it's width and height. This box is showing the target object in present frame of video.

Once we get our target object coordinate in starting frame, than we just need to keep track of it as video continues to play frame by frame. In order to achieve that first we calculate the back projection on the present video frame with the histogram model of target object. It will give us back projected image of frame. If we output the back projected image than we'll see the whole image is black except for the target object which are white. Further we give that image as input to MeanShift or CamShift algorithm. If we give input to MeanShift and it will return new starting coordinate of the box. It will show our target object in that video frame with its width and height value. The value of width and height of box it returns every time, will be same as one that we has given as input while detecting the target object in first step. This process will repeat till there are more frames in video.

After calculating MeanShift after every iteration we need to show the target object, which can be achieved be highlighting the target object with rectangle, and the coordinate of rectangle the one which we get as output from the MeanShift or CamShift algorithm.

4. Pros and Cons of Color based Object Tracking

Pros

1. This technique is based on color and not on features of target object so it require relatively very less computational power for detecting object from every frame.
2. In this method it is possible to get pretty good frame rate while detecting the object in video.
3. This method is easy to understand and implement, thus it is the best way for beginners to get started with machine learning and object tracking using OpenCV.

Cons

1. This method is not recognize the features of target object while detecting it from video so has very less accuracy.
2. This method will go in ambiguous state if there more than one objects in frame having same color as that of our target object.
3. If our target object goes out of frame than the method won't understand that there is no target object in frame, instead it will assume the other object whose histogram model matches the best with target histogram model as a target object.
4. Compare to deep learning algorithms our algorithm cannot understand context from past experience.

5. Scope of improvement

Now due to some requirement or just for curiosity, if we had to use the color based tracking than there are some techniques, tricks and tips that can be applied to get the better results.

Other than Back Projection there are other methods to scan the pixel pattern of target object in frame of video. We all know that in any situation if the noise is reduced from any sample than we can analyze if better. Same concept is applied here. That is if we remove the noise from the frame than we can analyze and detect the object better. Here this noise is low light image that is the bad image quality due to the low or insufficient light, in other words we are giving the threshold value to every pixel in order to accept it in input.

Above concept can be implemented using an in Build method from OpenCV library known as in Range function. Basic idea behind this function is to filter out the range of light from the frame. As an input we gives an image on which we want to apply the threshold, and then the range of color that we want to accept from image. Here the range is given in HSV format. In our case we will apply the range of color that we want to from image to be accepted. For every frame first we will apply the inRange method and filter out the pixel with low light which can potentially be recognized as noise. After this we'll apply the back projection with histogram model of target object.

So by this small trick we can see pretty good increase in accuracy of object detection and tracking but with only slight increase in the processing needs. HSV Color range can be found on online.

Another way to increase the accuracy is very obvious way to do so. There is a termination criterion which is required input for back projection, which defines the criteria to stop the algorithm and give the result. In these criteria we need to give epsilon value and max iteration, thus tweaking this two parameters we can improve the accuracy without much increase in processing needs.

6. Conclusion

Thus from the above discussion we can conclude that, OpenCV is easy to understand and quit powerful tool. Here we had discussed about the methods for tracking an object in any video. To recap first we make histogram for the target object than for small portions in every frame in video we'll compare histogram of target object with histogram of that object. We will continuously do same for each and every frame of video, along with highlighting where the histogram of target object match with the best in frame. This is how we keep track of an object in video.

But as we know the nothing is perfect in the world, thus there comes some flaws with this method. As this is color based tracking technique it can easy confuse the target object with other object with same color or relevant histogram. Also if video is playing and if target object isn't in the frame, than it can't understand that object was there and afterward it got out of frame or it wasn't in video from

beginning, that is it cannot understand context from past. But along with all these disadvantages there is a major advantage of using this method, which is need of computational power. This method can achieve pretty good frame rates on an average machine. So this method can be used to analyze the video footage where accuracy is second factor but time is the limiting factor.

Hence ever beginner can use this OpenCV with this method of tracking object to get the basics of object tracking without any deep mathematical understanding , also the OpenCV is an open source library hence no licensing is needed in order to use it which make the learning process even more enjoyable.

7. References

[1] https://docs.opencv.org/

[2] https://www.geeksforgeeks.org/

[3] *"GitHub - opencv/Opencv: Open Source Computer Vision Library". 21 May 2020.*

[4] Intel acquires Itseez: https://opencv.org/intel-acquires-itseez.html

[5] *"CUDA". Opencv.org. Retrieved 2020-10-15.*

[6] *Adrian Kaehler; Gary Bradski (14 December 2016). Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media. pp. 26ff. ISBN 978-1-4919-3800-3.*

[7] *Bradski, Gary; Kaehler, Adrian (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc. p. 6.*

[8] OpenCV change logs: http://code.opencv.org/projects/opencv/wiki/ChangeLog Archived 2013-01-15 at the Wayback Machine

[9] OpenCV Developer Site: http://code.opencv.org Archived 2013-01-13 at Archive. today

[10] OpenCV User Site: http://opencv.org/

[11] *"Intel Acquires Computer Vision for IOT, Automotive | Intel Newsroom". Intel Newsroom. Retrieved 2018-11-26.*

[12] *"Intel acquires Russian computer vision company Itseez". East-West Digital News. 2016-05-31. Retrieved 2018-11-26.*

[13] OpenCV: http://opencv.org/opencv-3-3.html

[14] OpenCV C interface: http://docs.opencv.org

[15] Introduction to OpenCV.js and Tutorials

[16] Cuda GPU port: http://opencv.org/platforms/cuda.html Archived 2016-05-21 at the Way back Machine

[17] OpenCL Announcement: http://opencv.org/opencv-v2-4-3rc-is-under-way.html

[18] OpenCL-accelerated Computer Vision API Reference: http://docs.opencv.org/modules/ocl/doc/ocl.html

[19] Maemo port: https://garage.maemo.org/projects/opencv

[20] BlackBerry 10 (partial port): https://github.com/blackberry/OpenCV