Tutorial on CAMShift Algorithm and How to use it using OpenCV and Python for Object Tracking

e-Yantra Team

July 10, 2016

Contents

1 Objective

The objective of this tutorial is to describe the working of "Continuously Adaptive Meanshift (Camshift) Algorithm" and to explain "How to use this algorithm in OpenCV and Python for Object Tracking"

2 Prerequisites

User should have handy knowledge of following before reading this tutorial.

- Basics of Python Language.
- Introduction to OpenCV.
- Basics of Image processing in OpenCV using Python.
- Knowledge on Mean Shift Algorithm,Histograms and Back projections.

3 Hardware Requirement

- A Computer with internal or external webcam.

4 Software Requirement

- Python 2.7.5 (with OpenCV and Numpy module)
- OpenCV 2.4.9
- numpy 1.7.1
- Any sample video(that can be used for object tracking)
- **Note :** These versions I had at the time of this tutorial.

5 Theory and Description

The full form of Camshift Algorithm is Continuously Adaptive Meanshift Algorithm and it is improved version of Meanshift Algorithm. So it is good to understand Meanshift Algorithm before Camshift algorithm to get better idea of Camshift algorithm.

5.1 Meanshift Algorithm

Meanshift is a non-parametric feature-space analysis technique, a so-called mode seeking algorithm. It is a procedure for locating the maxima of a density function given discrete data sampled from that function. In a sense, it is using a non-parametric density gradient estimation. It is useful for tracking the objects.

Moving objects are characterized by their color-histograms. Therefore the key operation of the object tracking algorithm is histogram estimation. Meanshift tracking algorithm is an iterative scheme based on comparing the histogram of the original object in the current image frame and histogram of candidate regions in the next image frame.

The aim is to maximize the correlation between two histograms.Object tracking for an image frame is performed by a combination of histogram extraction, weight computation and derivation of new location.

Consider you have a set of points as given in Figure 1. (It can be a pixel distribution like histogram backprojection). 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 Figure 1.

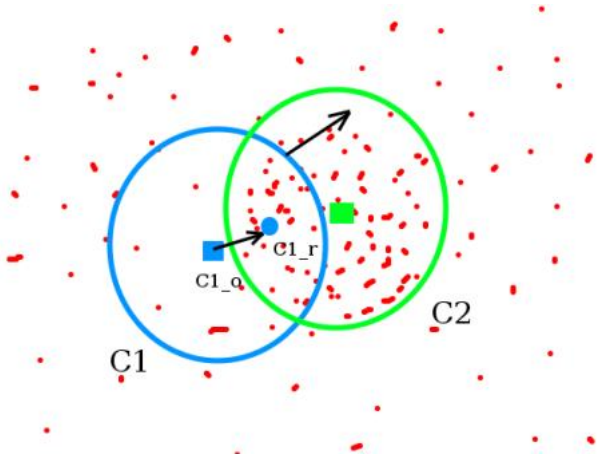


Figure 1: Meanshif Algortithm (Courtesy: opencv-python-tutroals.readthedocs.io)

The initial window in Figure 1 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 window. Surely they dont match. So move your window such that circle of the new window matches with previous centroid. Again find the new centroid. Most probably,it wont match. So move it again, and continue the iterations such

that center of window and its centroid falls on the same location (or with a small desired error). So finally what you obtain is a window with maximum pixel distribution. It is marked with green circle, named C2.

But the major drawback of meanshift algorithm is that it does not changes the size and orientation of the track window of the object with respect to scale and orientation of the object. It was removed in Camshift Algorithm, which is described below.

5.2 Camshift Algorithm

Camshift algorithm is an improvement of Meanshift algorithm known as Continously Adaptive Mean shift algorithm.It applies meanshift first.Once meanshift converges, it updates the size of the window using formula given in figure 2 and M00 is zeroth moment of track window.

$$s = 2 \times \sqrt{\frac{M_{00}}{256}}$$

Figure 2: Formula (Courtesy: opencv-python-tutroals.readthedocs.io)

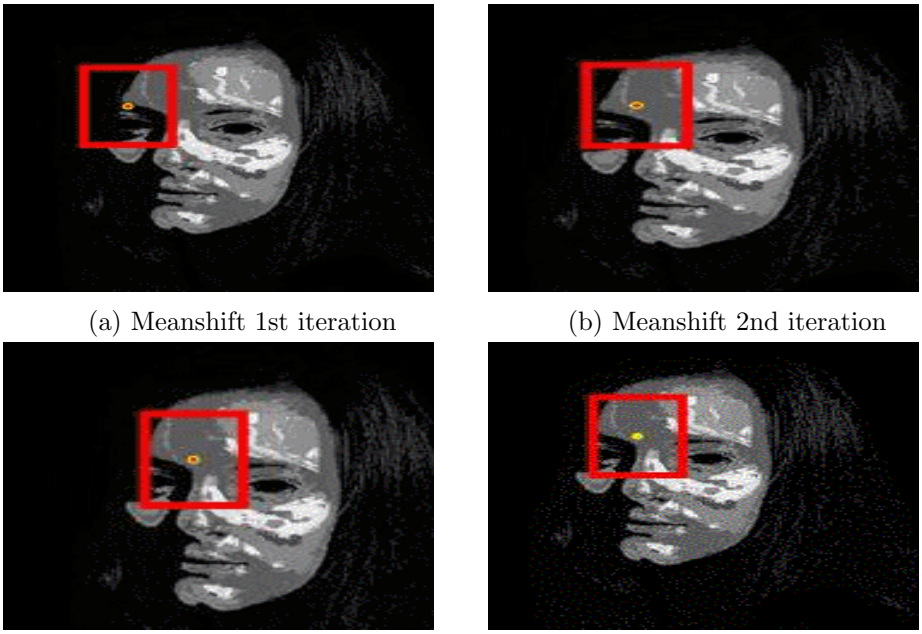
It also calculates the orientation of best fitting ellipse to it. Again it applies the meanshift with new scaled search window and previous window location. The process is continued until required accuracy is met.

It is almost same as meanshift, but it returns a rotated rectangle (that is our result) and box parameters (used to be passed as search window in next iteration).

The Camshift cleverly exploits the algorithm of mean-shift by changing the size of the window when it happened to convergence. The Camshift coupled is an adaptation to the color image sequences, and is operated in pursuit of real-time object.

5.2.1 Illustration of Camshift algorithm in a Second video frame

Let in first frame face was inside red rectangle (shown in Figure 3(a)) so it was selected as Region of Interest in first frame. And in Second frame, let face moves from previous position and comes the position as shown in Figure 3(a). So at first Camshift algorithm calculates mean of white pixels in back projection of frame. Then it makes mean as the new center of the window as shown, it is called meanshift iteration in Camshift Algorithm. Meanshift first iteration is shown in Figure 3(a). Meanshift second iteration , third iteration and convergence of Meanshift is shown in figure 3



(a) Meanshift 1st iteration

(b) Meanshift 2nd iteration

(c) Meanshift 3rd iteration

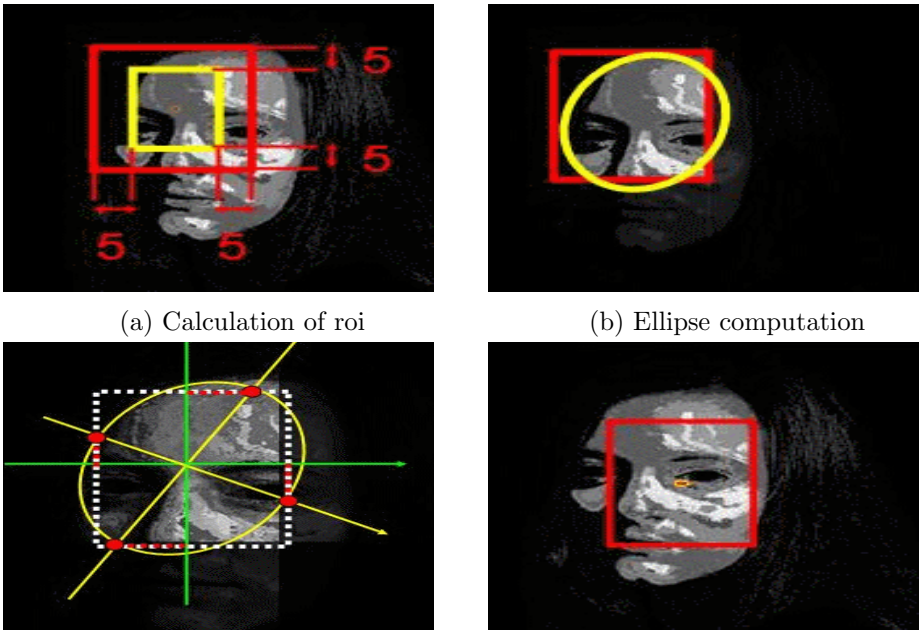
(d) Converged Meanshift

Figure 3: Courtesy: opencv-python-tutroals.readthedocs.io

Then it calculates size of the window according to formula given in Figure 2 as shown in Figure 4(a). Then if finds minimum area ellipse to give the orientation of object as shown in Figure 4(b). And this process continues iteratively as shown in Figure 4(c) and Figure 4(d).

After convergence it calculates converged ellipse as shown in Figure 5.

Camshift result is shown in Figure 6



(a) Calculation of roi

(b) Ellipse computation

(c) New meanshift axis

(d) Meanshift again

Figure 4: ellipse computation and applying Meanshift again (Courtesy: opencv-python-tutroals.readthedocs.io)

(a) Ellipse calculation

(b) Converged Ellipse

Figure 5: Converged ellipse calcualtion (Courtesy: opencv-python-tutroals.readthedocs.io)

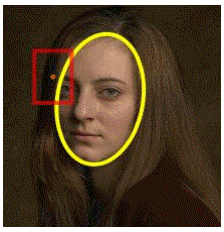


Figure 6: Resulting Camshift (Courtesy: opencv-python-tutroals.readthedocs.io)

5.3 Using Camshift Algorithm using Python OpenCV

To use Camshift Algorithm in OpenCV and Python following steps are required.

- After getting Region of Interest, we mask it for better result.
- We get 2D histogram of ROI using inbuilt OpenCV function cv2.calcHist().
- We normalize histogram in the range 0 to 1 using OpenCV function cv2.normalize.
- Initial ROI operattions is shown in Figure 7

```
# set up the ROI for tracking
roi = frame[r:r+h, c:c+w]

# convert it ROI to the HSV color space
hsv_roi =  cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)

# Masking hsv_roi for good results.
mask = cv2.inRange(hsv_roi, np.array((0., 60.,32.)), np.array((180.,255.,255.)))

# compute a HSV histogram for the ROI
roi_hist = cv2.calcHist([hsv_roi],[0,1],mask,[180,256],[0,180,0,256])

# normalize histogram
cv2.normalize(roi_hist,roi_hist,0,255,cv2.NORM_MINMAX)
```

Figure 7: Initial ROI operations

- Then in the next frame, color space is changed from RGB to HSV using OpenCV function cv2.cvtColor().
- It extracts object by back projecting ROI in current frame using ROI histogram.
- New bounding box, and minimum area rectangle is found using OpenCV inbuilt function cv2.CamShift() which uses back projection, previous bounding box, and termination criteria.
- code for frame operation is shown in Figure 8.

```
# convert the current frame to the HSV color space
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

# Apply backprojection on current frame with respect
# to roi histogram.
dst = cv2.calcBackProject([hsv],[0,1],roi_hist,[0,180,0,256],1)

# apply cam shift to the back projection, convert the
# points to a bounding box, and then draw them
ret, track_window = cv2.CamShift(dst, track_window, term_crit)
```

Figure 8: Frame operations

```
# apply cam shift to the back projection, convert the
# points to a bounding box, and then draw them
ret, track_window = cv2.CamShift(dst, track_window, term_crit)

# Draw it on image
pts = cv2.cv.BoxPoints(ret)
pts = np.int0(pts)
cv2.polylines(frame,[pts],True, [255,0,0], 2)
```

Figure 9: Drawing operations

- It draws rotating rectangle using output of cv2.CamShift(), 'ret'.
- Drawing of rectangle is shown in figure 9.
- This process will run iteratively.

6 Code

The Python code using OpenCV and Numpy can be found here. Step to use code are as shown below.

- Download and run code using Python Idle.
- press space bar to pause the video and give bounding box using mouse drag and drop.
- press space bar to track object.

7 Exercise

Real time object tracking in a sample video is shown in Figure 10 and Figure 11.

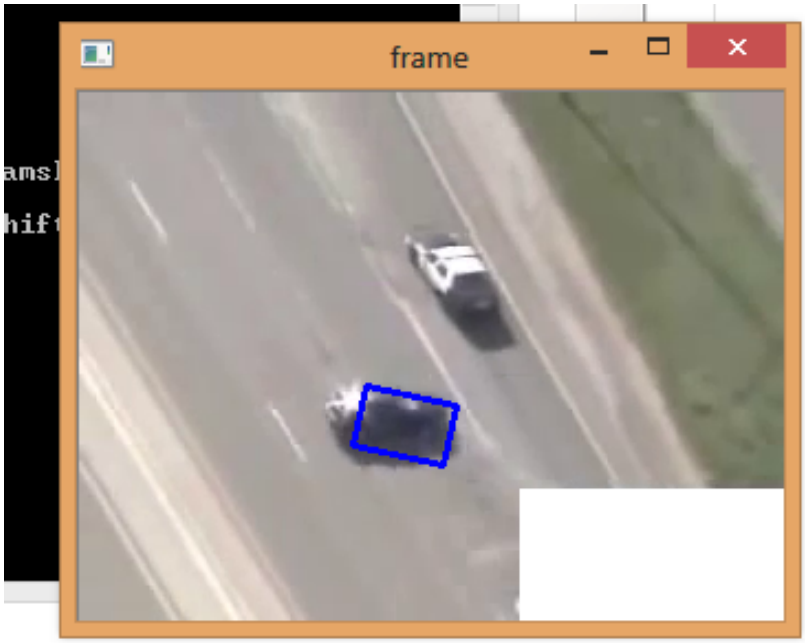


Figure 10: Selecting suspect car in a chase as ROI (Courtesy: youtube.com)

Figure 11: Tracking the suspect in the frame (Courtesy: youtube.com)

Real time object tracking in a webcam is shown in Figure 12 and Fig 13.

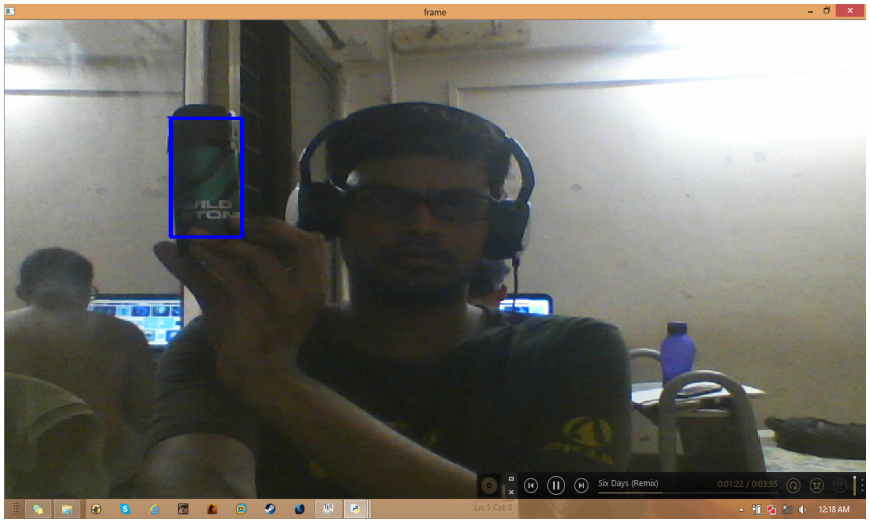


Figure 12: Selecting Deodrant as ROI

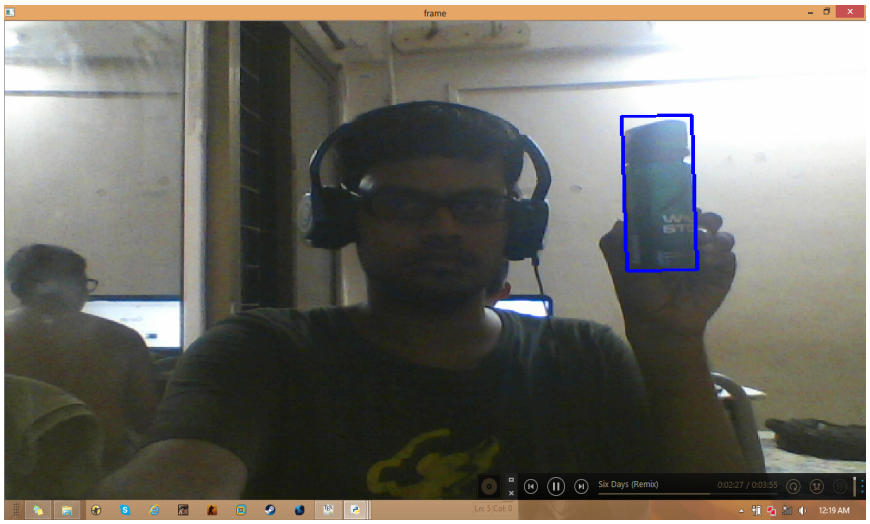


Figure 13: Tracking the Deodrant in the frame

8 References

1. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_meanshift/py_meanshift.html#meanshift
2. http://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_mean_shift_tracking_segmentation.php
3. http://www.computervisiononline.com/blog/tutorial-using-camshift-track-objects
4. https://sites.google.com/a/ualberta.ca/jinxin-he/programming/python/simpleobjecttracking
5. http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-wi