Object Tracking (Based on ROI)

e-Yantra Team

June 8, 2016

Contents

1 Object Tracking (Based on ROI)

The objective of this tutorial is to track an object(or some region) by giving a Region Of Interest(ROI)using a mouse to the module .We track the object/region by using algorithms like Meanshift and CAMshift techniques.

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

- Here we track the objects using color probability distribution.We consider the hue and saturation values(can also be done considering only hue value) and find their values in each pixel of the ROI .(taken from first frame)
- We track the objects by converting the ROI frame into Histograms and using back projection on the histogram we can vary the object from the background(like masking a colored object)

- After getting the back projection of the Object ,we use different algorithms like Mean and CAM Shift for tracking the object in the following frames

Figure 1: Sample Image

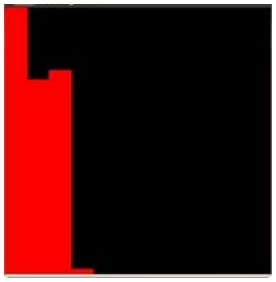


Figure 2: Histogram

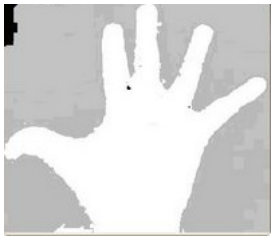


Figure 3: Back Projection

- Same process works in each frame and we track the colored object.
- **Note :** Camshift and Meanshift may not work properly when there is a similar colored object in the frame as we had taken hue and saturation values for tracking.

Algorithms used:

5.1 Mean Shift Algorithm

- Mean shift 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 detecting the modes of this density.
- Moving objects are characterized by their color-histograms. Therefore the key operation of the object tracking algorithm is histogram estimation. Mean-shift 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. (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 simple image given below:

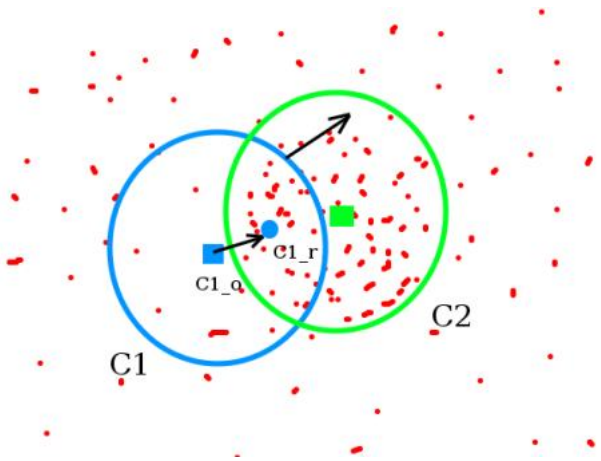
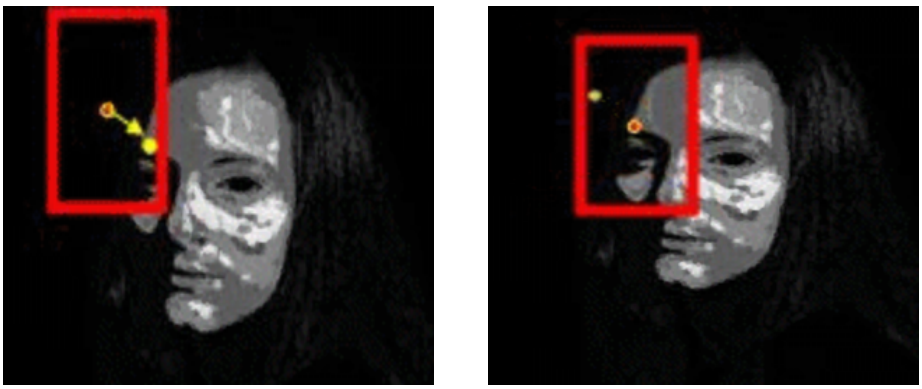


Figure 4

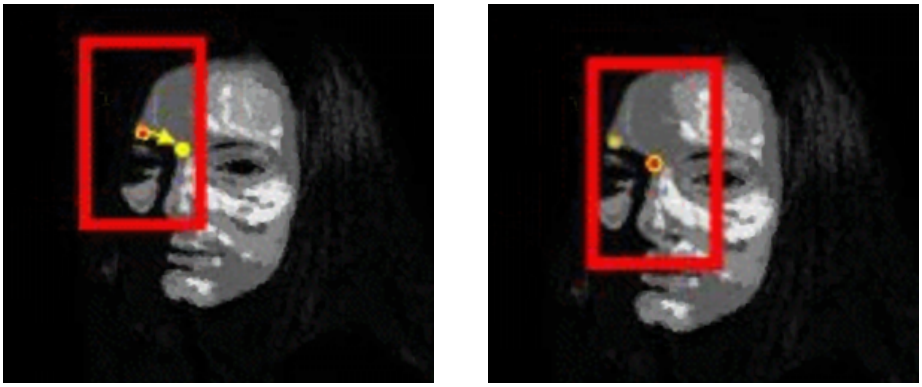
- The initial window is shown in blue circle with the name C1. Its original center is marked in blue rectangle, named C1.o. But

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 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.



(a) Mean calculation (b) Mean shift

Figure 5: Mean shift in first iteration



(a) Mean calculation (b) Mean shift

Figure 6: Mean shift in second iteration

(a) Mean calculation (b) Mean shift

Figure 7: Mean shift in third iteration

(a) Mean calculation (b) Mean shift

Figure 8: Mean shift in fourth iteration

Figure 9: Converged Image

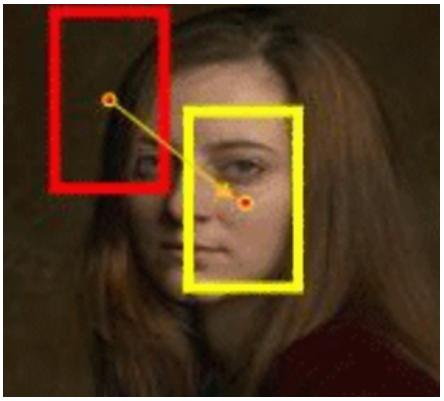


Figure 10: Shifted Window

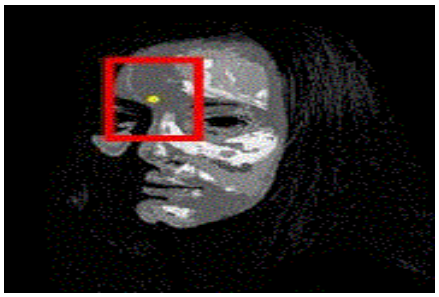
5.2 CAM Shift Algorithm

- CAM Shift is an improvement of Mean shift algorithm known as Continously Adaptive Mean shift algorithm.It applies meanshift first.Once meanshift converges, it updates the size of the window. 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.
- CAMSHIFT was implemented as such:
  - Initial location of the 2D search window was computed.
  - The color probability distribution is calculated for a region slightly bigger than the mean shift search window.
  - Mean shift is performed on the area until suitable convergence. The zeroth moment and centroid coordinates are computed and stored.
  - The search window for the next frame is centered around the centroid and the size is scaled by a function of the zeroth movement.
  - Reapeats again from step 2

(a) Meanshift 1st iteration

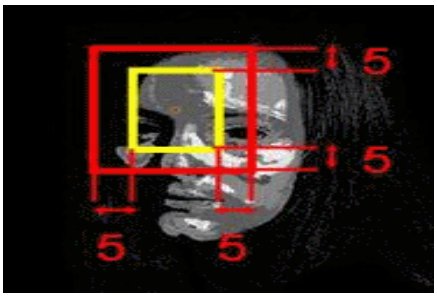
(b) Meanshift 2nd iteration

(c) Meanshift 3rd iteration

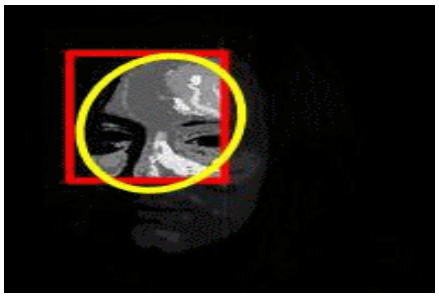


(d) Converged Meanshift

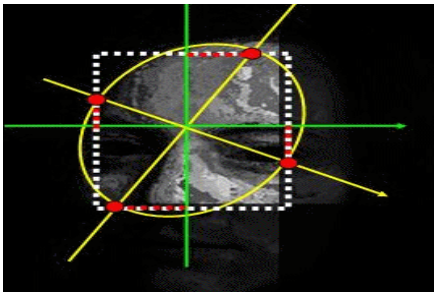
Figure 11: Meanshift



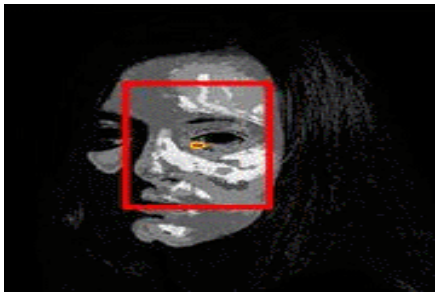
(a) Calculation of roi



(b) Ellipse computation

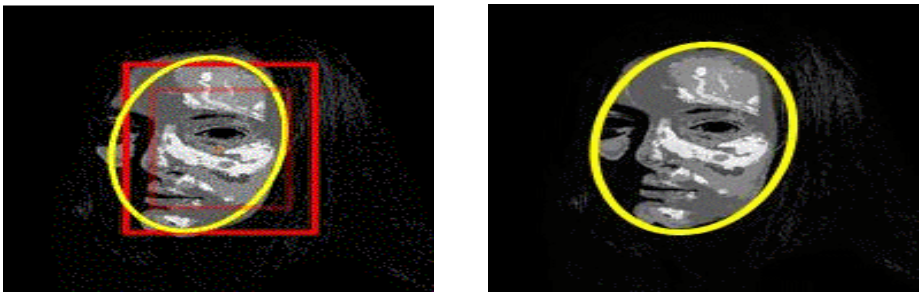


(c) New meanshift axis



(d) Meanshift again

Figure 12: ellipse computation and applying Meanshift again



(a) Ellipse calculation (b) Converged Ellipse

Figure 13: Converged ellipse calcualtion

Figure 14: Resulting Camshift

6 Experiment

The Python code using OpenCV and Numpy is given below.

```
# import the necessary packages
import numpy as np
import cv2

# (ix,iy) will be north west corner and
# (jx,jy) will be south east corner of ROI rectangle
ix,iy,jx,jy = -1,-1,-1,-1

# mouse callback function to select ROI in the frame
def select_ROI(event,x,y,flags,param):
```

```
    global ix,iy,jx,jy

    # If mouse left button is down,
    # set (ix,iy) = current co-ordinates
    if event == cv2.EVENT_LBUTTONDOWN:
        ix,iy = x,y

    # Else if mouse left button is up,
    # set (jx,jy) = current co-ordinates
    elif event == cv2.EVENT_LBUTTONUP:
        jx,jy = x,y
        # Draw rectangle using (ix,iy) and (jx,jy)
        cv2.rectangle(frame,(ix,iy),(jx,jy),(255,0,0),2)

# Grab the reference to the camera
cap = cv2.VideoCapture('sample.mp4')

# setup the mouse callback
cv2.namedWindow('frame',cv2.WINDOW_NORMAL)
# Binding select_ROI with the frame
cv2.setMouseCallback('frame',select_ROI)

# Decides to pause video or not to take ROI
pause = False

# Decides to track object or not
track = False

# keep looping over the frames
while(1):
    # grab the current frame
    ret ,frame = cap.read()

    # check to see if we have reached the end of the
    # video
    if ret == False:
        break

    # If pause is True, then go into ROI selection mode
    while(pause):
        # Show the current frame only
        # As frame is binded with select_ROI mouse callback function
        # So you can select ROI on this frame by mouse dragging
        cv2.imshow('frame',frame)
```

```
        # Press space bar after selecting ROI
        # to process the ROI and to track the object
        if cv2.waitKey(1) & 0xff == 32: #ascii value for spacebar

            # To prevent this loop to start again.
            pause = False

            # setup initial location of window
            r,h,c,w = iy , (jy-iy) , ix , (jx-ix)
            track_window = (c,r,w,h)

            # 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

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

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

            # Setup the termination criteria,
            # either 10 iteration or move by atleast 1 pt
            term_crit = ( cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT

            # Makes track = True to start tracking
            track = True

            # To terminate current loop.
            break

    # After ROI computation start tracking
    if track == True:

        # 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)

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

        # show the frame
        cv2.imshow('frame',frame)
        k = cv2.waitKey(20) & 0xff

        # If spacebar is pressed,
        # puase the frame to take ROI
        if k == 32: #ascii value for spacebar
            pause = True

        # If Escape key is pressed,
        # terminate the video
        elif k == 27: #ascii value for escape key
            break

## Releasing camera
cap.release()

## Destroy all open windows
cv2.destroyAllWindows()
```

6.1 Mean Shift Code

In Mean Shift we use the MeanShift function available in opencv.It is as follows:

```
(r, roiBox) = cv2.meanShift(backProj, roiBox, termination)
x,y,w,h = roiBox
cv2.rectangle(frame, (x,y), (x+w,y+h), 255,2)
```

7 Exercise

Real time object tracking in a sample video is shown below.

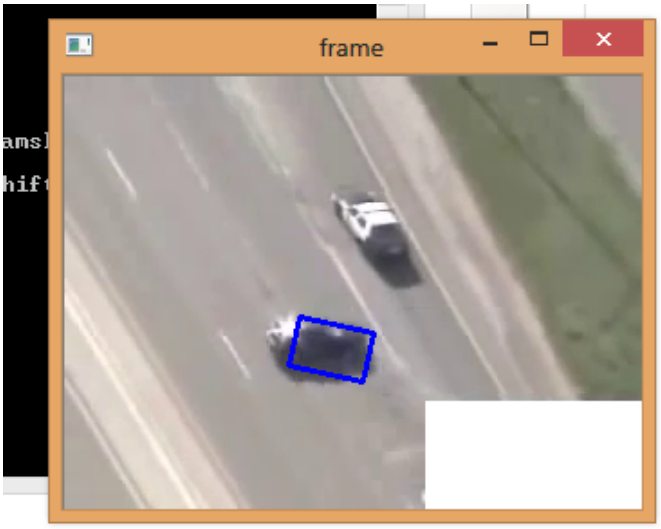


Figure 15: Selecting suspect car in a chase as ROI

Figure 16: Tracking the suspect in the frame

Real time object tracking in a webcam is shown below.

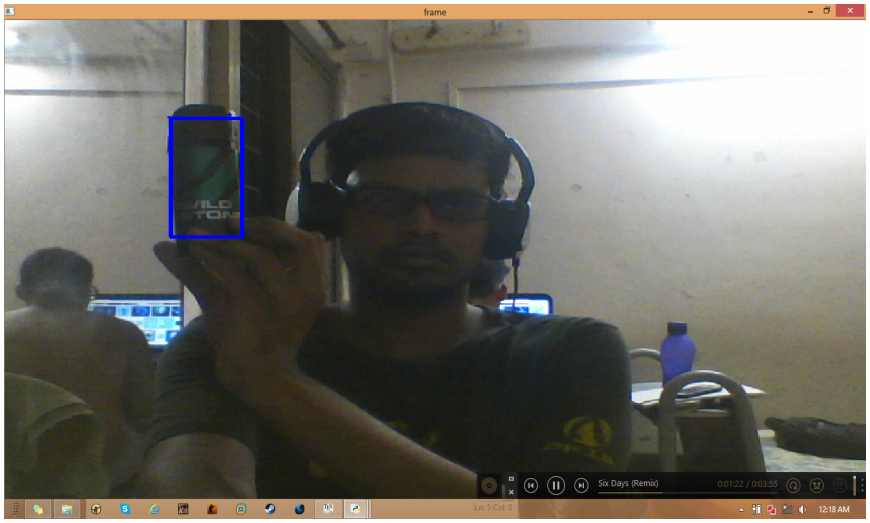


Figure 17: Selecting Deodrant as ROI

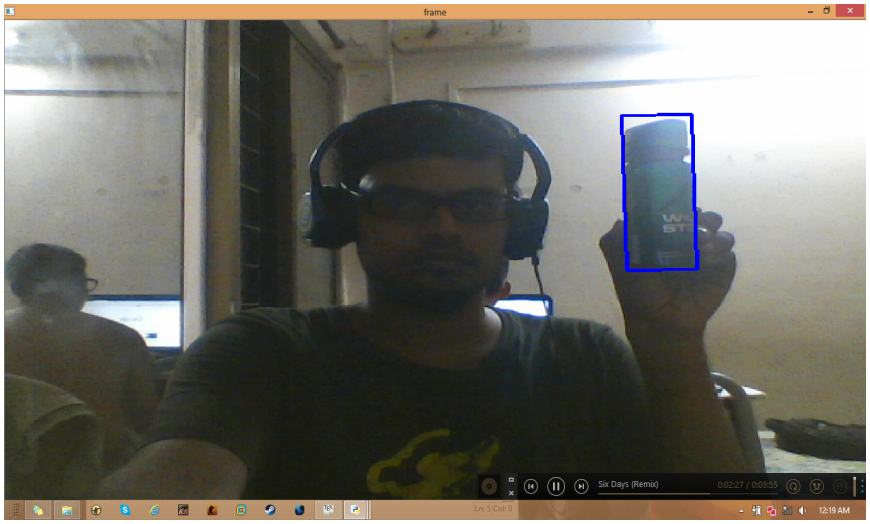


Figure 18: Tracking the Deodrant in the frame

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