# Report

## Mean Shift: Single Object Tracking in Images

**Name:** Keya Shukla

- **Problem Statement**

  Using OpenCV implement a single object tracker. Steps to be implemented:
  - a) Use a pre-recorded video or your webcam to have a video Capture object.
  - b) Mark the region of interest (ROI or the object you want to track) using it coordinates in the first frame.
  - c) Calculate the histogram of the ROI.
  - d) Iteratively calculate the histogram at each location (using cv2,calcBackProject) and then apply mean shift to get the updated location of the ROI.

- **Prerequisites**

  - Software:
    - Python 3 (Use anaconda as your python distributor as well)

  - Tools:
    - Numpy
    - OpenCV

  - Dataset: A pre-recorded video

- **Method Used**

  Mean shift is a non-parametric feature-space analysis technique for locating the maxima of a density function, a so-called mode-seeking algorithm. Application domains include cluster analysis in computer vision and image processing.

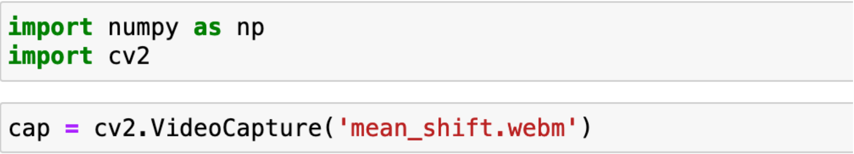
  The mean shift algorithm can be used for visual tracking. The simplest such algorithm would create a confidence map in the new image based on the color histogram of the object in the previous image, and use mean shift to find the peak of a confidence map near the object's old position. The confidence map is a probability density function on the new image, assigning each pixel of the new image a probability, which is the probability of the pixel color occurring in the object in the previous image.

  Mean shift is a procedure for locating the maxima—the modes—of a density function given discrete data sampled from that function.

Although the mean shift algorithm has been widely used in many applications, a rigid proof for the convergence of the algorithm using a general kernel in a high dimensional space is still not known.

- **Implementation:**

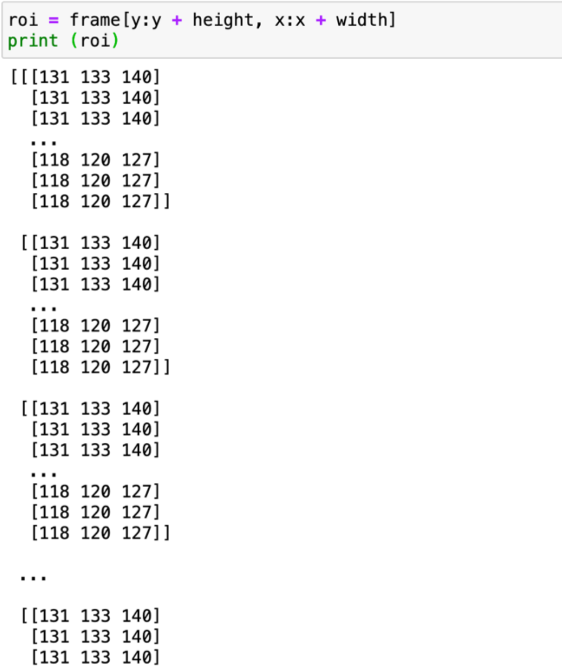
**1.** Load all required libraries



```
import numpy as np
import cv2
```

```
cap = cv2.VideoCapture('mean_shift.webm')
```

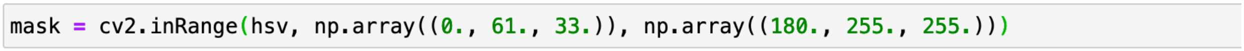
**2.** Tracking ROI



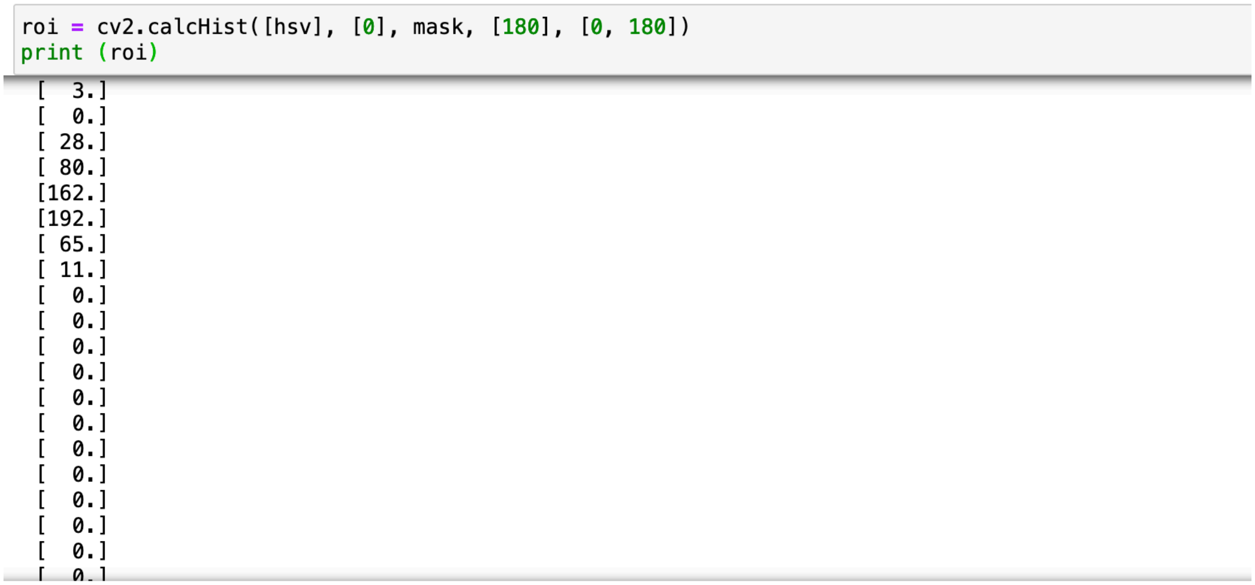
```
roi = frame[y:y + height, x:x + width]
print (roi)
```

```
[[[131 133 140]
  [131 133 140]
  [131 133 140]
  ...
  [118 120 127]
  [118 120 127]
  [118 120 127]]

 [[131 133 140]
  [131 133 140]
  [131 133 140]
  ...
  [118 120 127]
  [118 120 127]
  [118 120 127]]

 [[131 133 140]
  [131 133 140]
  [131 133 140]
  ...
  [118 120 127]
  [118 120 127]
  [118 120 127]]

 ...

 [[131 133 140]
  [131 133 140]
  [131 133 140]
```

**3.** Applying mask over ROI



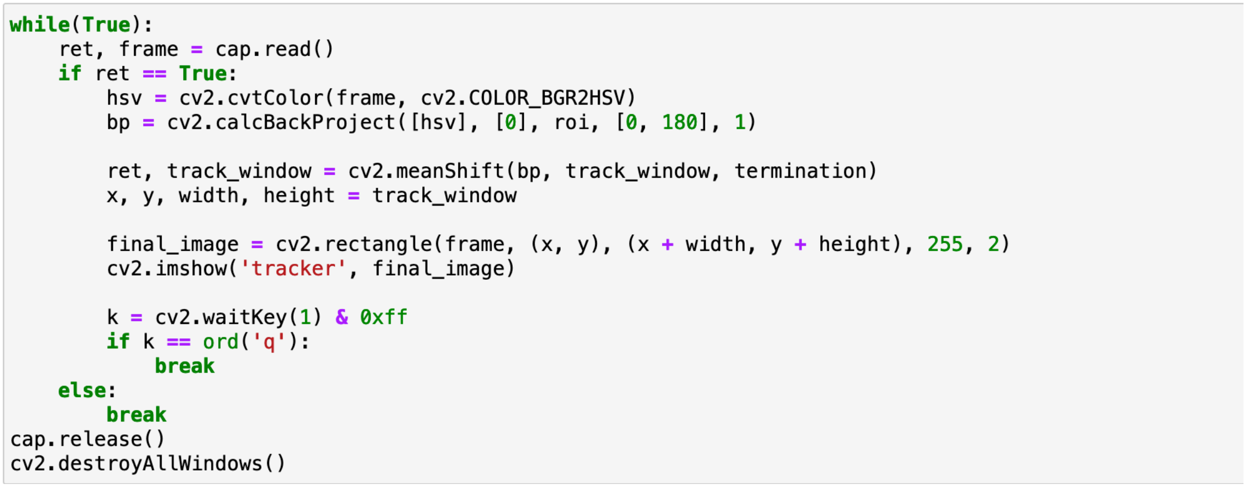
```python
mask = cv2.inRange(hsv, np.array((0., 61., 33.)), np.array((180., 255., 255.)))
```



```python
roi = cv2.calcHist([hsv], [0], mask, [180], [0, 180])
print (roi)
```

```
 [  3.]
 [  0.]
 [ 28.]
 [ 80.]
 [162.]
 [192.]
 [ 65.]
 [ 11.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
```

## 4. Implementing Mean Shift



```python
while(True):
    ret, frame = cap.read()
    if ret == True:
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        bp = cv2.calcBackProject([hsv], [0], roi, [0, 180], 1)

        ret, track_window = cv2.meanShift(bp, track_window, termination)
        x, y, width, height = track_window

        final_image = cv2.rectangle(frame, (x, y), (x + width, y + height), 255, 2)
        cv2.imshow('tracker', final_image)

        k = cv2.waitKey(1) & 0xff
        if k == ord('q'):
            break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```

- **Results:**

For result, it is required to watch the video presentation.