Master 2 AIC : Image Mining
Lab 3 : Object Tracking in Videos

Corentin LELOUP - Wafa BOUZOUITA

Introduction

In this practical work, we aim to understand the challenges and difficulties of object tracking in videos, to experiment and develop solutions based on Mean Shift and Hough transform algorithms.

1 Mean Shift

Q1

The mean shift algorithm is an iterative process which computes the mean shift value for the current position and then moves the point to its mean shift value. The process of computing mean shift iterates until it fulfills a certain convergence condition and it is limited by the fixed kernel bandwidth.

Among the strengths of the mean shift algorithm we have that it is simple to implement, does not require data training as it is robust against outliers. The computational complexity of mean shift algorithm can be written as $O(Tn^2)$, where T is the number of iterations and n is the number of data points. Or in our problem, we want to capture the mug which is not constituted with many pixels and T in only equal to 10. So, the Mean shift algorithm convergence is rapid.

Among the weaknesses of the mean shift algorithms we have that we need to use adaptive window size because improper window size can lead modes to be merged. The window size (bandwidth selection) is not trivial to choose.

Q2

The Mean Shift method needs an initial conversion of the existing RGB-color-based video in the HSV color space, and a back-projection histogram in the HSV color space is used to obtain the color information of the object region that is to be tracked. With HSV, a pixel is represented by 3 parameters : Hue, Saturation and Value. However, unlike RGB, HSV does not use the primary color to represent a pixel. Instead, it uses hue, which is the color or shade of the pixel. The saturation is the intensity of the color, where a saturation of 0 represent 0 and a saturation of 255 is maximum intensity. Value tells how bright or dark the color is. In our experiments, we evaluated the given Mean shift algorithm on the mug tracking problem. This algorithm uses only the Hue component, where this mug is colored by white and black pixels. Or, transforming pixels of the frame from RGB color to HSV color, by considering only the Hue component make the mug a little bit invisible, whereas the hand that takes the mug is orange, it reminds visible after the transformation. So, an idea seems good to resolve this problem consists in considering Value and Saturation components, which increases the precision of tracking the mug but the stability decreases.

Before modifying the code, we visualize in the following figures, the distribution (dst) and the frame 81. We clearly see in the dst image that the hand is constituted with pixels of high value compared to the mug.

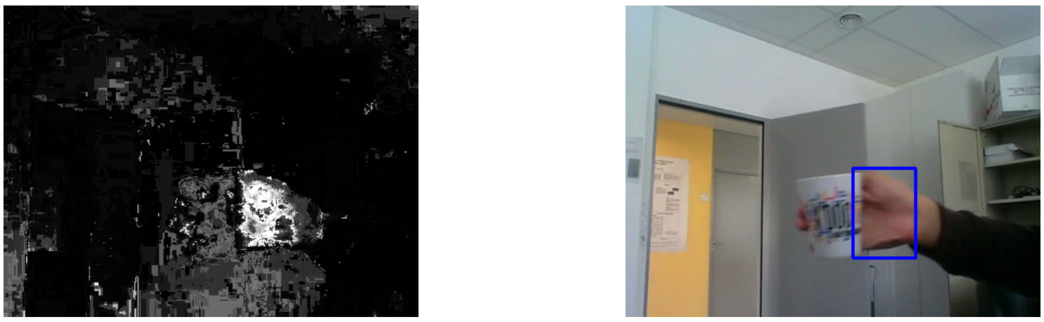


Figure 1: Dst image (left) and original image (right).

To make the hand value pixels less higher, we changed the interval of hue component from [0,180] to [15,180]. We obtained the following figure :

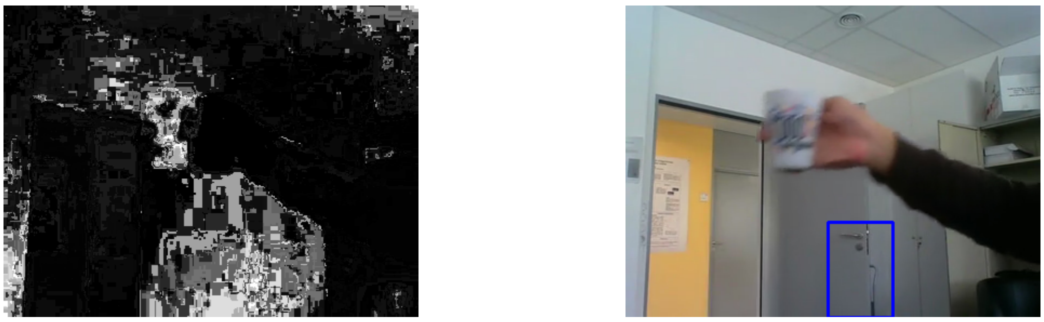
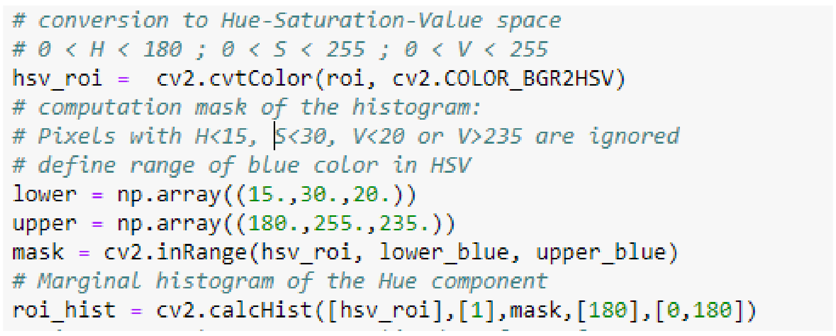
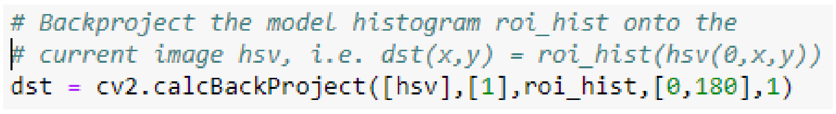


Figure 2: Dst image (left) and original image (right).

In this case, the value pixels of the door is higher then the mug. So, to ameliorate the result, we used the Saturation component instead of Hue component to calculate the histogram. The final modified code is written as follow :


```
# conversion to Hue-Saturation-Value space
# 0 < H < 180 ; 0 < S < 255 ; 0 < V < 255
hsv_roi =  cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
# computation mask of the histogram:
# Pixels with H<15, S<30, V<20 or V>235 are ignored
# define range of blue color in HSV
lower = np.array((15.,30.,20.))
upper = np.array((180.,255.,235.))
mask = cv2.inRange(hsv_roi, lower_blue, upper_blue)
# Marginal histogram of the Hue component
roi_hist = cv2.calcHist([hsv_roi],[1],mask,[180],[0,180])
```


```
# Backproject the model histogram roi_hist onto the
# current image hsv, i.e. dst(x,y) = roi_hist(hsv(0,x,y))
dst = cv2.calcBackProject([hsv],[1],roi_hist,[0,180],1)
```

Finally, we obtained better results. The mug was well tracked.

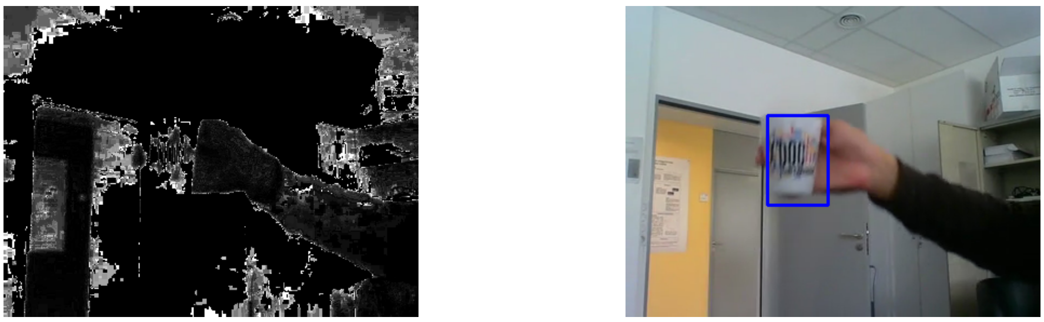


Figure 3: Dst image (left) and original image (right).

In futur work, we can also ameliorate our algorithm performances by adapting window size with size and rotation of the target. The solution is called

CAMshift (Continuously Adaptive Meanshift) published by Gary Bradsky in his paper "Computer Vision Face Tracking for Use in a Perceptual User Interface" in 1988.

2 Hough Transform

Q3

To obtain the pixels whose gradient is interesting, we computed the spetial derivative with the classical [-1,0,1] convolutionnal mask in 4 of the 8 directions of the hough transform (the other 4 being deduced by taking the opposite of the first case). The 4 matrixes used are the following :

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We then keep only the pixels whose gradient values are above a chosen threshold (here we chose 100) for each direction. Based on this, we can plot on the original image with its high gradient value points highlighted for each direction. The results are visible on the Figure 4 on a test image. It is of course possible to plot the same image for other gradient directions or other thresholds.

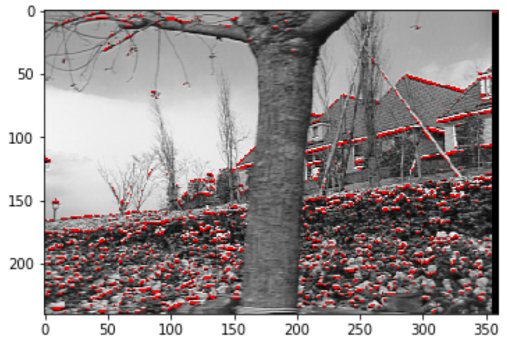


Figure 4: Pixels with a gradient value above threshold in direction 0

Q4

We store in a dictionnary the pixels of interest obtained in the previous question by the vector of their relative position to the center. The obtained R-table is used to create a map of votes on a new image. The point which gets the largest amount of votes is the new center of the rectangle of interest.

The performances are good, the model allows efficient tracking on the mug video, however some irregularities appear when the mug appears bigger or smaller on the video. The computing time is significantly longer than with the mean shift which might result in the system being unusable in a real-time context despite its increased performance.

One can see on Figure 5 that the tracking follows effectively the mug an not the hand like in the previous version with the Meanshift.

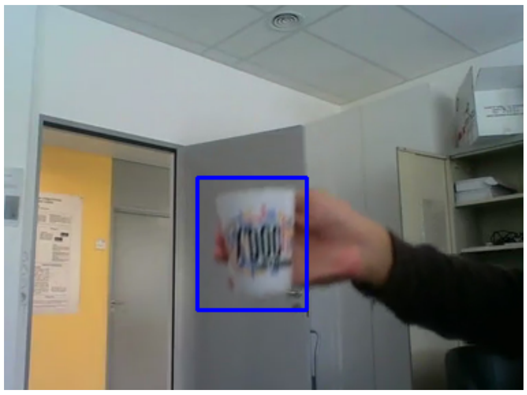


Figure 5: Random frame of the video

Q5

By replacing the exhaustive search of the maximum by the local search by meanshift, the performance doesn't seem to be affected but the computation time is greatly increased. To adapt the tracking to an object whose side is varying, one should use another algorithm like for example CAMshift which updates the rectangle of interest size after the meanshift and applies meanshift again with the new ROI.