Image Analysis: Mean Shift

Oscar E. Ramos Ponce - Mohammad Ali Mirzaei
Université de Bourgogne
M.Sc. in Computer Vision and Robotics

I. Introduction

Tracking is a very widely used technique that finds a variety of applications, ranging from military such as anti-aircraft and missiles shelter defense systems to very commercial cases like unmanned autopilot navigation, monitoring and security. The main concern in all the application is finding the object of interest, which is in general called target, and then trying to follow the object by tracking the target by the vision system. Hence, the first step is to localize the object and then to detect it in the proceeding frame and to calculate the motion vector. Background subtraction is a very robust method to localize the object in the very initial frame. After that, the histogram model is used as a feature to estimate the motion vector. By having this motion vector the bounding box is moved to a new place to indicate the motion of the target and to localize the position of that target. For color images we use a color model to extract this feature but the method does not change so much. In order to cope with different size of target we need to find the scale and rescale the tracking system. In the following brief report we will try to answer all this issues and reflect back to the matlab code which is provided for this purpose.

II. Preparation of the data for the algorithm

A. Initialization

First the images are read, and they are stored in a 3 dimensional matrix since in Matlab it is particularly fast to perform operations along the different dimensions of the arrays. In this way, the 3 dimensional matrix has the form $\mathbb{R}^3 \rightarrow \mathbb{R}$ (in reality the real numbers are reduced to integers). The two first components of the domain $x \in [0, H]$, $y \in [0, W]$ correspond to the pixel location (starting from the top left corner), and the third one $t \in [1, N_{img}]$ to the image sequence number (associated with the time), where H represents the height of the image, W its width and N_{img} the total number of images in the sequence. The range is reduced to the set of gray levels $I(x, y, t) \in [0, 255]$.

B. Background subtraction

In order to obtain the background from the set of given images, the median of each pixel in time was performed, profiting the facts that there is no motion in the camera or background from image at time t to image at time $t+1$, and that the objects that do not constitute the background are moving along the scene and appear in different positions in time, that is, the moving object appears only for a short period of time in a given location. Even though this assumption is restrictive, it is accurate for the used dataset. Then, the background image for a certain (x, y) pixel position, $I_{back}(x, y)$, will be given by

$$I_{back}(x, y) = med\{I(x, y, 1), I(x, y, 2), ..., I(x, y, N_{img})\}$$

where *med* represents the median operation. The result obtained is shown in Figure 1.

Figure 1: Background obtained

C. Region of Interest

Since we are interested in tracking the trajectory of the car, the region of interest is precisely the car in the first image, and we want to draw a bounding box around it. For this sake, first, the absolute value of the difference between the first image and the background was computed as:

$$I_{diff}(x, y) = |I(x, y, 1) - I_{back}(x, y)| \quad (1)$$

The image I_{diff} was then binarized applying a threshold to it. The result after the thersholding is shown in Figure 2 (left). However, there are some undesired elements that need to be eliminated. It can be noticed that these unwanted elements have a small number of constituting pixels, then, a criteria for their elimination was to first label the connected elements using 8-connectivity, leading to several connected regions, and then keeping only the region with most connected elements, that is, the largest region. This region constitutes the desired region of interest ROI and corresponds to the car. Even though it is not necessary, to improve the result, the thresholded image can be first closed applying successively a dilation and then an erosion. Figure 2 (right) shows the result after keeping only the largest region.

The maximum and minimum positions of this region were determined and used as the limits of the bounding box of the ROI. The widht w_r and height h_r of the region

Figure 2: Original thresholded image (left), after selecting the largest region (right)

were computed, as well as its center (x_c, y_c), which is the mean of the extremes in x and in y. The bounding box is shown in Figure 3.

Figure 3: Bounding box on the first image of the sequence

III. Basic Mean-Shift algorithm

The algorithm described here corresponds basically to [1], [2]. First, the grey levels for each one of the pixels in the image patch have to be quantized into m bins. For $m = 8$ bins, the function that has to be applied is similar to the one shown in Figure 4, assuming that initially the levels ranged from 0 to 255. Let the number of pixels in

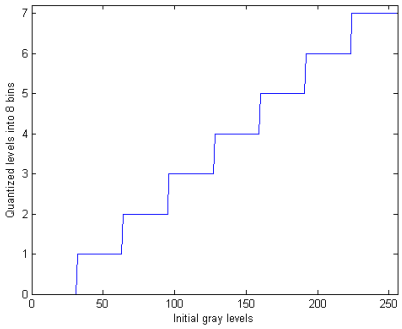

Figure 4: Quantization function for 8 bins

the ROI (patch) be n. For a certain pixel $X_i = (x_i, y_i)$ in the ROI, where $i \in [1, n]$, its quantized value or bin number, will be $b(X_i)$. A new image containing only this values for the ROI, at any time t, was created as:

$$b(X_i) = b(x_i, y_i) = \lfloor \frac{I(x_i, y_i, t)}{256/m} \rfloor \tag{2}$$

where $\lfloor f \rfloor$ represents the floor function (largest integer not greater than f). Using this formulation, $b(X_i)$ will have values ranging from 0 to $m-1$. The target model q is a discrete m-bin histogram obtained from the ROI in the first image and it contains m elements: $q = [q_1, q_2, ..., q_m]$. Each element is given by [1]:

$$q_{u+1} = C \sum_{b(X_i)=u} k\left(\| \frac{X_i - X_c}{h} \|^2 \right) \tag{3}$$

$$q_{u+1} = C \sum_{i=1}^{n} k\left(\| \frac{X_i - X_c}{h} \|^2 \right) \delta(b(X_i) - u) \tag{4}$$

where $u \in [0, m-1]$, X_c are the coordinates of the center of the ROI, C is a normalization factor so that q sums up to 1, h is half of the size of the window, and k is the Epanechnikov kernel defined by

$$k(x) = \begin{cases} 0.5 C_d^{-1}(d+2)(1-x) & , x < 1 \\ 0 & , x > 1 \end{cases} \tag{5}$$

Here, we are dealing with a 2 dimensional space, so $d = 2$ and $C_d = \pi$, the volume of the unit 2 dimensional sphere (a unitary circle). Note that the subindex of q is $u+1$ instead of u. This change in the notation was necessary since the values of the histogram $b(X_i)$ start with 0, whereas the elements of q, q_i start with 1.

For the computation, first, the center of the ROI X_c is calculated, the values of q are initialized to zero and then the pixels belonging to the ROI are read in order. For each pixel, the square of the centered and normalized magnitude is computed, in this case as:

$$\| \frac{X_i - X_c}{h} \|^2 = \left(\frac{x_i - x_c}{h_r/2} \right)^2 + \left(\frac{y_i - y_c}{w_r/2} \right)^2 \tag{6}$$

where w_r and h_r are the width and height of the region, as described previously. This value is the argument of the k function. Then, the value of k is added to the element of the target model q_{u+1} considering that $b(X_i) = u$. Finally, after the whole computation, q is normalized to sum up to 1, like a probability distribution.

For the patch in the next frame (the target candidate), the color model is computed exactly in the same way, but now it is named $p(Y_c)$ instead of q. The Y_c is the center of the new patch and refers to the fact that now the central point of the patch is going to be changed in order to find a good similarity with the Target Model. Each one of the pixels in the new frame is called $Y_i = (x_i, y_i)$. To evaluate how this new color model matches the previous one, the Brattacharyya coeficient ρ is calculated as:

$$\rho(q, p(Y_c)) = \sum_{u=1}^{m} \sqrt{q_u p_u(Y_c)} \tag{7}$$

The weight for each pixel Y_i in the ROI is then computed as:

$$w_i = w(Y_i) = \sum_{u=1}^{m} \delta(b(Y_i) - u) \sqrt{\frac{q_u}{p_u(Y_c)}} \tag{8}$$

which basically states that for each pixel, the root square is taken only for the component of p and q that corresponds to the value of $b(Y_i)$, all the other other components are

neglected by the δ function. After this weighting factor, the new location Z is:

$$Z = \frac{\sum_{i=1}^{n} Y_i w_i g(\|(Y - Y_i)/h\|^2)}{\sum_{i=1}^{n} w_i g(\|(Y - Y_i)/h\|^2)} \quad (9)$$

The function $g(x) = -k'(x)$ and in this case, since the kernel $k(x)$ depends linearly on x its derivative is a constant and can be omitted from the previous equation since it appears in both the numerator and denominator and will be then canceled after the division. However, the fact that it is different to zero only when $x < 1$ has to be taken into account. After obtaining the new position of Z inside the patch, with reference to its center, it has to be translated to the position in the image, which can be easily done by adding it to the coordinates of the center of the patch in the image frame. Doing this process iteratively yields to obtain the new center of the patch, which corresponds to the desired center of the tracked objecet. Figure 5 shows the evolution for the second image, where the original patch of the first image can be seen at the left and the new computed patch for the second image is shown at the right. As observed, the patch has been moved automatically to a new more convenient position to fit better the model, for this particular image, it has moved to the left, since the motion of the car was in that direction.

Figure 5: ROI for the second image: patch corresponding to the first image (left), new patch after the mean-shift algorithm (right)

Summarizing, the basic algorithm is as follows:

1) Compute the target model using (3) or (4)
2) Initialize the patch for the new frame in the position of the previous frame, Y, and obtain its model $p(Y)$
3) Compute the distance ρ given by (7).
4) Calculate the new location Z applying the mean shift, using (9).
5) Compute the new values of $p(Z)$ and rho.
6) If the new distance is smaller than the previous one, that is $\rho[p(Z), q] < \rho[p(Y), q]$, change Z to $0.5(Y + Z)$.
7) If $\|Z - Y\|$ is small, stop, otherwise assign Z to Y and go to step 2.

In order to accelerate the velocity of the program, sacrificing the accuracy, the loop can be controlled for a number of fixed iterations before it reaches a small difference in the last step. This approach leads to bounding boxes slightly less localized in the image, but the tracking still works and is faster. The result of the tracking for different positions of the object is shown in Figure 6. It can be observed that the size of the patch does not change from frame to frame, but the size of the car decreases as the car goes further in the image. Nevertheless, the algorithm is still able to track the car, even though the size of the new car is smaller than the initial one.

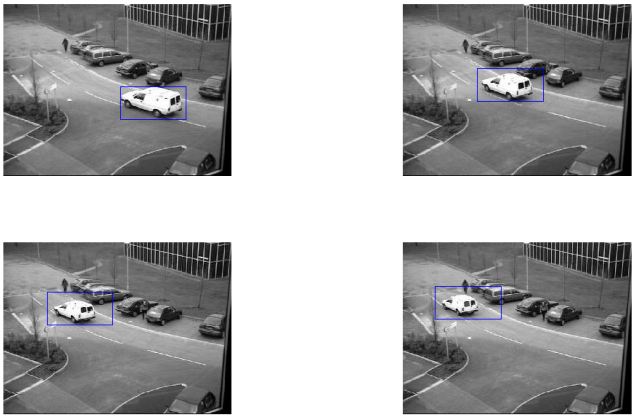

Figure 6: Results for different times using the same ROI size

IV. Adaptation to handle the varying size of the object

The simplest way is to define different scales and then compare the size of current bounding box with the previous one to find the scale down/up between two consecutive windows. But the best way to do is using γ (gamma) distribution function which called γ-normalization method [3]. The γ-normalization function is the following:

$$f_\gamma(\theta, V) = |V|^{\gamma/2} f(\theta, V) \quad (10)$$

where θ represents the mean and γ the variance. We can observe that this function is directly linked with the position of the local patches which is defined by mean. Moreover we can deduct the shape by the variance and by having two consecutive variances it is easier to estimate the scale. Some may say that using Gaussian is easier and we should use that distribution. However, we prefer the gamma distribution because it is more robust in practice [3].

V. Tracking with color images

For tracking with color images, the easiest way is to change the color image to gray scale and then apply the previously described method. Another, more interesting approach, is to use a color histogram distribution, instead of a gray level distribution, for both the target model and the target candidate. In fact, we are applying a model for a color space and then we extract the color histogram, which is a 3D structure where each one of its dimensions corresponds to a certain discretized color plane. Then, we will have a 3D matrix with m_1 x m_2 x m_3 size, where the size of each dimension depends on the number of bins that we are using for that specific dimension. However, it is more usual to define a square matrix.

The result for color images depends on how the color distribution is modeled. Using an alternative color model,

possibly by changing the color space, can improve the tracking results for the plane sequence.

We assume that the shape of a non-rigid object is approximated by an ellipsoidal (rectangle or circular or any other kind of shape) region in an image. We select the region exactly in the same way that we selected in the very initial steps for the initial condition. Let x_i denote a pixel location, θ the initial location of the center of the object in the image, V the variance, M the number of bins in the histogram and let $b(x_i)$ be a function that assigns the color value for each pixel to its bin. The value of the m^{th} bin is calculated by

$$o_m = \sum_{i=1}^{N_{v_0}} \mathcal{N}(x_i; \theta_0, V_0)\delta[b(x_i) - m] \tag{11}$$

In this equation $\mathcal{N}$ represents the Gaussian distribution and δ is the Kronecker delta function. The weighting coefficient calculation is exactly the same one as that was used in the first part, the farther, the lesser weight and the nearer the larger weight.

Improvement: In order to increase the accuracy and robustness of the algorithm, a Kalman filter is introduced in the body of the algorithm and is used to estimate the parameters so that we can be sure that we are going in the right direction and selecting the right region. The main idea of this improvement comes from [3].

VI. Limitations of the algorithm

From what we saw in the implementation, this algorithm is quite sensitive to the initial condition, which means that if we locate the bounding box an approximate place, the bounding box will go to a wrong place after some movement of the object, but not so far away, though. Second, in case that the difference between the background and the object is small in the sense of illumination and contrast, it is hard to track the object with the color model that was defined. One of the possible ways is to try to adapt the color space according to the case of study, if we are looking for high accuracy. But in case we have good difference between the object and its background it works well due to a lot of redundant information.

VII. Conclusions

In conclusion, this method, mean shift tracking, works well when there is a lot of information in the frames and a notorious difference in contrast and illumination without presence of noise. In fact, this algorithm is good for classical paper study work, but in dayly-life experimental applications it might present some problems when facing occlusions, for example, and one of the best alternatives are the particle and kalman filter. These recent methods have a good performance in real applications even with the presence of the noise. A combination of mean-shift tracking with kalman or particle filter would yield even to better results.

References

[1] D. Comaniciu and V. Ramesh, "Real-time tracking of non-rigid objects using mean shift," Jul. 8 2003, uS Patent 6,590,999.

[2] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[3] Z. Zivkovic and B. Krose, "An EM-like algorithm for color-histogram-based object tracking," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1. IEEE, 2004.