University of Burgundy

Visual Tracking

Lab 2

Mean-shift

Author:
Di MENG

Supervisor:
Pro. Désiré Sidibé

November 05, 2017

1 Introduction

The objective of this lab is to grab the knowledge of Mean-shift algorithm and implement it in Matlab for visual tracking application. When aiming at moving object tracking, Mean-shift considers feature space as an empirical probability density function(pdf) and locates the maxima of the pdf. This report explains the main steps of the algorithm and shows the tracking results based on the given image sequence. The posed questions are answered at the end.

2 Algorithm and Implementation

2.1 Tracker initialization

Before tracking the target in the sequences, the position of the target should be initialized. There are two ways to initialize the target: 1)use a detection method 2)initialize the target manually. In this lab, we use the second method. Matlab function 'imcrop' is used to crop the ROI(region of interest). Then the ROI is extracted as the target by the following function in the code:

imPatch = extract_image_patch_center_size(I, c, w, h)

Where I is the image. c is the center of the extracted patch. w and h are the width and height of the target respectively.

2.2 Object model

The appearance of the object is represented as a color distribution. The color distribution is in format of discrete histogram. To decrease the computation, the histogram has several bins such as 8 or 16 instead of 256 bins. What's more, since the histogram does not take the spacial distribution into account. We introduce a kernel function to weight the contribution of pixels according to their distance from region's center. Thus, the probability distribution which is also the object model can be obtained by:

$$q(u) = C \sum_{i=1}^{n} k(\|x_i\|^2)\delta(b(x_i - u))$$

Where $\|x_i\|^2$ is the distance from pixel x_i to the region center(the distances are normalized). And the kernel function used is:

$$k_E(x) = \{ \begin{array}{ll} \text{(1-x)}/2\pi & \text{if x<1} \\ 0 & \text{if x} \geq 1 \end{array}$$

When calculating the value of each bin in the implementation, instead of plus one, value k is added to accumulated the bins' value. The function to realize it in the codd is:

cd = color_distribution(imagePatch, m)

Where m is the number of bins and output is the color distribution of the target which is the object model.

2.3 Color density matching

Using the function from last section, the models of initial frame and current frame are obtained. The aim of this algorithm is to track the patch with largest similarity. So we use the Bhattacharyya coefficient to measure the similarity.

$$\rho[p,q] = \sum_{u=1}^{m} \sqrt{p(u)q(u)}$$

Where p and q are the object models obtained from the last section. ρ is the level of matching. The larger the ρ, the more matching the models. The function of this step in the code is:

k = compute_bhattacharyya_coefficient(p,q)

2.4 Weight computing

Because Mean-shift treats points as a probability density function, dense regions in feature space corresponds to local maxima or modes. The center of the tracking target is moving towards to the denser location. So computing the weights of pixels in the patch is necessary. The weights can be computed by:

$$w_i = \sum_{u=1}^{m} \delta(b(y_i) - u)\sqrt{\frac{q_u}{p_u(y)}}$$

The function implemented in the code is:

weights = compute_weights(imPatch, qTarget, pCurrent, Nbins)

The size of the weights is as same as the patch's.

2.5 New location

Mean shift is applied in this step. Since we are given the patch and the weights of all the pixels. The new location can be computed as:

$$z = \frac{\sum_{i=1}^{n_h} y_i w_i}{\sum_{i=1}^{n_h} w_i}$$

Where y_i represents each pixel position in the current window. The function in the code is:

z = compute_meanshift_vector(imPatch, weights)

2.6 Tracking

For each frame, we can use the previous target position to extract models and estimate the new location. By comparing the similarity of target model and current model using Bhattacharyya coefficient, the new location can be updated and obtained.

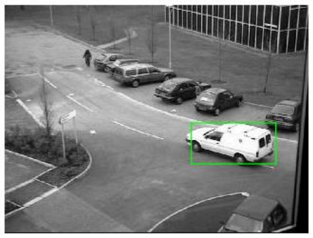
3 Results

The algorithm is tested with the given car sequences. The result is shown in Figure 1. We can see from the figure, the Mean-shift tracking algorithm works well. And the algorithm is efficient with less computation time.

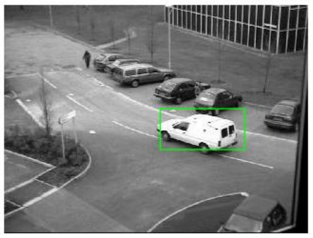
4 Improvements

4.1 Varying size of object

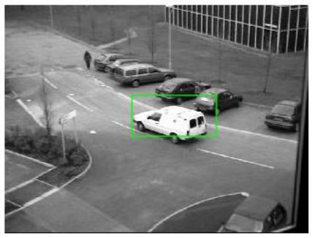
The algorithm can be improved with varying object scales. A 10% scale adaptation can be involved in the algorithm. Instead of computing the model of a fix-size patch, two more models can also be computed and considered by adding and subtracting 10% of the size. Moreover, we can add one scale variable and apply Mean-shift algorithm in 3 dimensions.



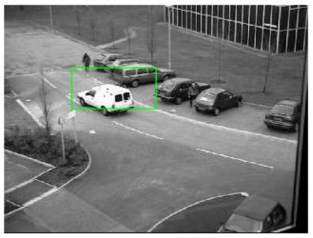
(a)



(b)



(c)



(d)

Figure 1: Tracking car in the sequence using Mean-shift algorithm

4.2 Color images

The algorithm can also be improved to track color images. In the step of computing color distribution, we computed one histogram with one channel. This function can be updated to output three histograms according to three channels from color image. And when comparing the Bhattacharyya coefficient, a vector can be compared rather than a value.

4.3 Limitations

The limitation of the Mean-shift algorithm is that inappropriate window size selection can influence the result of tracking. In order to deal with occlusion problem, feature extraction can be used to represent better model. What's more, Kalmen filter can be applied for more robust tracking with occlusions.