MEAN-SHIFT, OBJECT TRACKING

Helena Cots, Paula Castillo

ABSTRACT

The problem of object tracking in real time has been studied for a long time. Its goal is to segment a region of interest where a moving object is detected and keep a track of its position in each frame. It can be solved with a sequence of steps where a good approach of target representation and localization is the clue to achieve good results. The representation is done by a feature histogram and the localization consists on the gradient-based optimization: mean-shift algorithm. That is, the target localization problem can be solved using the criterion of the local maxima.

Keywords - Object tracking, target representation and localization, Bhattacharyya Coefficient, smooth similarity function, distance minimization.

1. INTRODUCTION

The definition of object tracking is the process of locating a moving object in real time. It is an essential task in many computer-vision applications such as human-computer interaction, security and surveillance, augmented reality, traffic control, driver assistance, and medical imaging.

The main idea is to associate targets in consecutive frames, where the target model is the object in the previous frame and the target candidate would be the localization of the object on the next frame of the video. Ideally, repeating this process on each frame of the video.

There are many obstacles when tracking an object: its motion might be random or faster than the frame rate, there could be partial or full occlusion of the object we are trying to track, the illumination and the background can affect when evaluating the region of the object, among many other problems.

The main ideas on the state-of-the-art object tracking algorithms for target representation and localization could be distinguish as the Kernel-Based Tracking (mean-shift tracking) [1], an iterative localization process based on the local maxima of a similarity measure (Bhattacharyya coefficient), and Contour Tracking (active contours or Condensation algorithm) [2] , the detection of object's boundaries.

The first one, the mean-shift algorithm, is a non-parametric feature space analysis technique for locating the maxima of a density function. Its applications on machine learning and data mining can be for instance supervised learning, clustering, structured prediction, *tracking* and smoothing.

In case of object tracking, it computes the maximum similarity between the target candidate and the target model in order to decide if the target candidate will be the model for the following frame. Moreover, due to the assumption of considering only small changes in the location and appearance of the target in two consecutive frames, would increase the efficiency of gradient-based localization steps.

2. MEAN-SHIFT ALGORITHM

As explained before, a target model must be chosen before computing the mean-shift algorithm: choose a feature space and represent the model by its probability density function (q) in this feature space. We define the location of the target model as y0, and the location of the target candidate on the subsequent frame, **y**, which it is also represented by its pdf (p(y)). Before computing the both pdf estimated from the data it is used the kernel profile k(x), which assigns smaller weights to pixels farther from the centered location. In order to reduce complexity, we will use m-bins histograms.

It is important to consider that the histograms are computed from the pixels inside the ellipse of the target, for this reason it is important to establish its scale h centered at its location.

2.1. Target model

As we know where the first target model is, we can manually determine the target position (the center pixel y0) and pre-define the ellipse size [hx,hy] in the initial frame.

Having all the normalized pixel locations inside the ellipse defined as $\{x_i\}_{i=1...n}$ in the region of the target model (using *getPointsInEllipseBorder* and *getPointsInEllipse* functions) and the previous parameters, we can compute the probability of the feature $u = 1...m$ in the model:

$$q_u = C \sum_{u=1}^{n} k(||x_i||^2)\delta[b(x_i) - u] , \qquad (1)$$

where n is the number of locations inside the ellipse and u the number of the bin. The b function gives the pixel at the location x_i the index of the bin of the histogram where it belongs, and C function is a normalization constant.

2.2. Target candidate

Once the target model is computed, the target candidate will be evaluated, at first, at the same location (y0) of the target model: p(y0) (histogram of the new model). The implementation of the mean-shift allows the algorithm to iterate through all the possible locations of the new target using similarity, until it ends with the final position. In practise, the algorithm does as much 20 iterations to find that new center y.

$$p_u(y) = C_h \sum_{u=1}^{nh} k(||(\mathbf{y}-x_i)/h||^2)\delta[b(x_i) - u] \qquad (2)$$

In this case, it is important to define *h* as it is the bandwidth that defines the relation within the pixels of the target candidate (it is necessary to adapt the ellipse).

2.3. Weights computation

The objective of this computation, is giving more importance to the locations where values are more similar to the target model, which will help the algorithm to find where the object (target model) in the new frame is. The weights will only have a value different from 0 when $b(x_i) = u$, that is that the bin assignment to location x_i corresponds to the bin u.

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{q_u}{p_i(y_0)}}\ \delta[b(x_i) - u], \qquad (3)$$

The more similar the target candidate to the target model in the bin is, the more weight will have when having to compute the new location y1. [3]

2.4. New location

The following step is to compute the new location $\hat{y}_1$ corresponding to the target in the current using the values of the weights obtained in the step above. To find this location, the distance should be minimized as a function of y (gradient descent). As mentioned before, the process starts from the position of the target model and searches on its neighbourhood. The assumption that has to be made is that there is no drastic changes in location neither appearance of the object.

It consists on iterating over the weights through which we achieve to predict the random movement of the object.

$$\hat{y}_1 = \frac{\sum_{i=1}^{n_h} x_i * wi}{\sum_{i=1}^{n_h} wi} . \qquad (4)$$

2.5. Stopping criteria

The way we determine if the algorithm has to stop or not is through a threshold. So, if $||\hat{y}_1 - y_0|| < \varepsilon$, we stop iterating. If not, we should return to *weights computation* step.

2.6. Possible improvements

In order to enhance the object tracking algorithm we could have implemented also two other steps before computing the stopping criteria:

1. Giving the Bhattacharyya Coefficient:

$$\rho\ [\hat{p}(y), \hat{q}] = \sum_{u=1}^{m} \sqrt{p_u(y) * q_u} . \qquad (5)$$

   Evaluate ρ at the new location $\hat{y}_1$ through checking the similarity between the previous y_0.

2. While ($\rho\ [\hat{p}(y_0), \hat{q}] < \rho\ [\hat{p}(y_1), \hat{q}]$) , y_1 is updated as $y_1 = ½\ (y_0 + y_1)$ and evaluate again $\rho\ [\hat{p}(y_1), \hat{q}]$.

The stopping criterion threshold is the result of the constraint that the vectors y1 and y0 should be near the same pixel in original image coordinates. The improvement mentioned above, is only to avoid numerical potential numerical problems in the

maximization on the algorithm due to the Bhattacharyya Coefficient (5). After trying to implement this steps as it is seen in the matlab code, we have seen that it was no necessary to compute this coefficient because, as many experiments has proved, there is no huge improvement on the algorithm.

3. EXPERIMENTAL RESULTS

When we implemented the steps 1 and 2 of the *Possible improvements*, we realized that it could not work well in our computation and, as the code by itself work perfectly without the improvements, we chose to discard this implementations.

However, we tried to change the number of maximum iterations and the value of the threshold on the stopping step.

With 20 iterations max, 0.5 as a threshold and the y0 of the target model [71,61] we obtain the following next locations on some of the consecutive frames:

As it was mentioned before, the new locations should be around the previous one of the target model as it cannot be an abrupt movement.

With a lower threshold, like 0.001, surprisingly it has no negative effect on the algorithm, even it has the same performance as before. Despite the tracking of the object is well done, the locations of the new targets are not the same as with the ancient threshold. By the other hand, putting a large threshold (20) gave another target locations but, as before, the tracking was good. Maybe we had luck this with this object, but it might not work that well in other videos.

If we try to change the maximum number of iterations to 4 (as it is done in real time), the result of tracking is the same as before but with different target locations between the first and the last frame. This means that in this video there is no high complexity on tracking the object: the motion is smooth and there is no changes on the histograms.

4. CONCLUSIONS

Although our method works as expected, it has some limitations. For example, in real time the number of iterations are much more lower than in our case (among 4). Another limitation is that it cannot be used when there are abrupt changes in motion neither huge changes on the object appearance.

As it is said before, one of the most important facts to create an object tracking methods with successful results is to focus on the target representation and localization. For instance, there is an article where it is explained how they dealt with the issue of target representation. They used a model based on chaos theory [4] with which they achieved brilliant results that seems to be better than state-of-the-art tracking algorithms under rotation, illumination and scale changes.

5. REFERENCES

[1] Dorin Comaniciu, IEEE, Visvanathan Ramesh and Peter Meer. “Kernel-Based Object Tracking”. IEEE Transactions On Pattern Analysis and Machine Intelligence. Vol. 25, No. 5, May 2003.

[2] Frank Dellaert, Wolfram Burgard, Dieter Fox, Sebastian Thrun. “Using the CONDENSATION Algorithm for Robust, Vision-based Mobile Robot Localization.”

[3] Adnan Munawar. Mean-shift-matlab. https://github.com/adnanmunawar/mean-shift-matlab/blob/master/getWeights.m

[4] Marjan Abdechiri, Karim Faez, Hamidreza Amindavar, Eleonora Bilotta, *"Chaotic target representation for robust object tracking, Signal Processing: Image Communication"* Volume 54, 2017, https://doi.org/10.1016/j.image.2017.02.004.