MSc VIBOT & Computer Vision

Visual Tracking

Practice

Désiré Sidibé
dro-desire.sidibe@u-bourgogne.fr

Tracking a Moving Object II: Mean-Shift

Introduction
The goal of this *Visual Tracking* module is to learn about, and, more importantly, to learn how to use basic tracking algorithms and evaluate their performances.
After *Background Subtraction*, we will now use a very effective technique for real-time tracking called *Mean-Shift*. Mean-Shift is a deterministic algorithm, as opposed to probabilistic ones, which solves the data association problem (matching) in a very effective way.

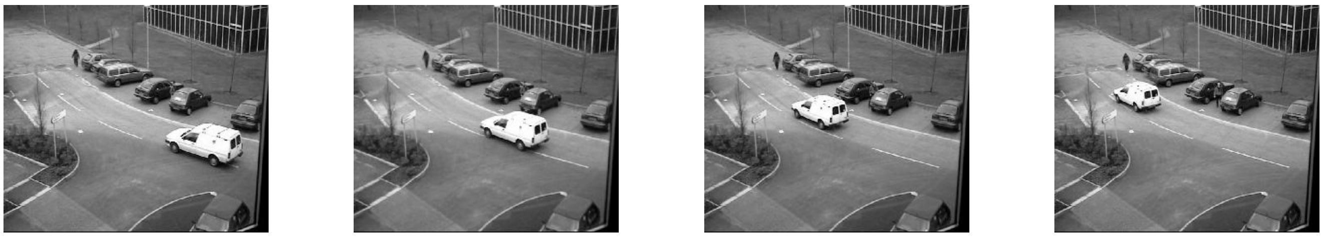

Figure 1: The car sequence.

Tracker initialization
We need to initialize the tracker with the position of the target (in the above figure it is the car) in the first frame. For this task, we can

- use a dtection method
- initialize the target manually

We can for example use a background subtraction technique (Practice session 1) to initialize the tracker. Once we have the initial position of the target, we can to track it through the sequence.

Mean-Shift tracking
Mean-Shift (MS) is widely known as one of the most basic yet powerful tracking algorithms. Mean-Shift considers feature space as an empirical probability density function (pdf).
If the input is a set of points then MS considers them as sampled from the underlying pdf. If dense regions (or clusters) are present in the feature space, then they correspond to the local maxima of the pdf.
For each data point, MS associates it with the nearby peak of the pdf. On abstract level, the MS algorithm can be summarized as follows:

1. Fix a window around each data point.

2. Compute the mean of data within the window.
3. Shift the window to the mean and repeat untill convergence.

NOTE:
Because MS treats points as a probability density function, dense regions in feature space corresponds to local maxima or modes. For each data point, it performs gradient ascent on the local estimated density until convergence. The stationary points obtained via gradient ascent represent the modes of the density function. All points associated with the same stationary point belong to the same cluster.

The algorithm in detail
1. Object model
The first step of the algorithm is to represent the appearance of the object as a color distribution. We will describe the object's color distribution by a discrete m-bins histogram (try $m = 8$, $m = 16$), and we will call this distribution the object's color model q.
Let $\{x_1, \ldots, x_n\}$ be theset of pixels in the region corresponding to the bounding box of the object, and $b(x_i)$ denotes the color bin at point x_i, i.e. $b()$ is a function which assigns to each pixel one of the m bins of the histogram.
A normal histogram h would be defined as

$$h(u) = C \sum_{i=1}^{n} \delta(b(x_i) - u), \quad \text{for} \quad u = 1, \ldots, m;$$

where where C is a normalizer (so that the value in the histogram sum to one) and $\delta(.)$ is the function:

$$\delta(a) = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}$$

But this representation does not take into account the spatial arrangement of pixels, that is two differents images can have the exact same histogram. Therefore, we introduce in the histogram representation a kernel that weights the contribution of pixels according to their distance from the region's center. So our object's model is a probability distribution q given 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 we use the Epanechnikov profile defined by

$$k_E(x) = \begin{cases} \frac{1}{2} C_d^{-1} (d+2)(1-x) & \text{if } x < 1 \\ 0 & \text{otherwise} \end{cases}$$

and C_d is the volume of the unit d-dimensional sphere. In our case, $d = 2$.

2. Color density matching
The second step of the algorithm is to find the objet in a new frame of a sequence. To do this, we have to look around the previous position of the object, and each such potential position defined a target candidate.
Each candidate is also represented by a color distribution p, and we evaluate the similarity between the model and candidate distributions, usingthe *Bhattacharyya coeficient* ρ:

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

ρ can be seen (and it is in fact, equals to) the cosine of the angle between the two vectors $[\sqrt(p_1), \ldots, \sqrt(p_m)]^T$ and $[\sqrt(q_1), \ldots, \sqrt(q_m)]^T$.
For each frame, we wish to find y that maximizes ρ. This y is the location of the target. Hence, large ρ means good color match.

3. Tracking algorithm
Given the distribution $\{q_u\}$ of the model and the location y in previous frame

1. Initialize target location in current frame as y

2. Compute $\{p_u(y)\}$ and $\rho[p(y), q]$

3. Apply mean-shift: compute new location z as

$$z = \frac{\sum_{i=1}^{n_h} y_i w_i g(\left\|\frac{y-y_i}{h}\right\|^2)}{\sum_{i=1}^{n_h} w_i g(\left\|\frac{y-y_i}{h}\right\|^2)},$$

   where n_h is the number of pixel in the region centered at y, $g = -k'$, and the weights are given by

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

4. Compute $\{p_u(z)\}$ and $\rho[p(z), q]$

5. While $\rho[p(z), q] < \rho[p(y), q]$, do $z \leftarrow \frac{1}{2}(y + z)$

6. If $\|z - y\|$ is small enough, stop; Else, set $y \leftarrow z$ and goto (1)

The tracking consists in running for each frame the algorithm describe above. Thus, given the target model, the new location of the target in the current frame is found to minimize the distance between the two distributions p and q in the neighborhood of the previous location.

What do do?
Q1: Implement a mean-shift algorithm to track a signle object in a sequence.
What can be done to deal with the varying size of the object? (You can refer to the original Mean-Shift paper)

Q2: How would you modify your code to deal with color images? Apply your tracking algorithm to the given sequences and any sequence of your choice. Comments?

Q3: What are the limitations of this algorithm?
How can we deal with occlusion? Demonstrate it.

NOTES:
1. Many codse can be found online, so I shall check submissions for similarities. You are free to look online for reference, but please make sure the code you submit is original!
2.A Matlab skeleton code is provided for help.