Kernel Based Tracking
(Multivariate Density Kernel)

Advanced Image Analysis - Caleb Assenhøj

Introduction (motivation)

The Kernel based tracker is a visual tracker that mainly is meant for *"Target Representation and Localization" in real-time*. It is able to track non-rigid objects, recorded with a normal camera rate (24-30 fps), on a colored RGB scene. It is robust to partial occlusion, clutter, distractors, camera motion and blurring [1] (of course the tracker has its limit inside the scope of these scenarios). It even works relatively robust without a background model/segmentation or a motion model, but of course these two properties would improve the tracker. Therefore it has special attraction in the field of surveillance. Extensive amounts of research has been performed based on the method described in the original publication [1], primarily to enhance the tracking as will be discussed in the improvement section.

Definition

Target Representation

First a feature space is chosen. Then we define a reference *"target model"*, that is represented by its pdf q (target representation) in the feature space. This reference model could e.g. be the color pdf of the target and is represented by a circle region in the given frame which is spatial centered in zero (this is a simplification of the original reference model described in the publication [1] which has an ellipsoidal region that is being normalized to a unit circle for different target dimensions). In the next frame, a *"target candidate"* is defined at location y and is defined by the pdf $p(y)$. Both pdfs estimates the data. We estimate the pdfs in a efficient way by using discrete densities, which is computed from a m-bin histogram (normalized histogram).

The Kernel

We choose a Epanechnikov $K_E(x)$ kernel because the kernel's weights corresponding to pixels further away from the center, are less important and are weighted less. This is due to possible occlusions and interference with background or other similar boundary effects.
The target region from the target model is registered with a convex and monotonically decreasing kernel K with a profile $k(x)$ defined as:

$$k(x): [0, \infty] \to \mathcal{R}, \quad K(\mathrm{x}) = k(\|\mathrm{x}\|^2)$$

Target Model & Target Candidate Definition

The target model is defined:

$$\hat{\mathrm{q}} = \{\hat{q}_u\} = \sum_{i=1}^{n} \frac{k(\|\mathrm{x}_i\|^2)\delta(b(\mathrm{x}_i)-u)}{\sum_{i=1}^{n} k(\|\mathrm{x}_i\|^2)}, \quad \sum_{u=1}^{n} \hat{q}_u = 1$$

The target candidate is defined:

$$\hat{\mathrm{p}}(\mathrm{y}) = \{\hat{p}_u(\mathrm{y})\} = \sum_{i=1}^{n_h} \frac{k\left(\left\|\frac{\mathrm{y}-\mathrm{x}_i}{h}\right\|^2\right)\delta(b(\mathrm{x}_i)-u)}{\sum_{i=1}^{n_h} k\left(\left\|\frac{\mathrm{y}-\mathrm{x}_i}{h}\right\|^2\right)}, \quad \sum_{u=1}^{n} \hat{p}_u = 1$$

Where u is the features and kernel size is defined by the h which can be adjusted adaptively, as will be explained in the section improvements.

Similarity Measure

We define a similarity measure based on the Bhattacharyya coefficient, which measures the overlap between two statistical samples or populations. The Bhattacharyya coefficient is being used as a correlation score between the target model step and the target candidate step. It has clear geometric interpretations, uses discrete densities and approximate a chi-square statistic, which is a good similarity or dissimilarity measure. This similarity function produces a smooth result, which makes the gradient based optimization inherited in the kernel based tracker much faster [1]. In the case of tracking, the similarity coincides with the likelihood that a tracked target already located in the previous frame is present in a y position in the current frame.
The Bhattacharyya coefficient is defined by:

$$\hat{\rho}(\mathrm{y}) \equiv \rho[\hat{\mathrm{p}}(\mathrm{y}), \hat{\mathrm{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathrm{y})\hat{q}_u}$$

Target Localization

In order to find the location corresponding to the target in the current frame, the Bhattacharyya , must be maximized as a function of y. The location procedure starts from the position in the previous frame (target model) and searches in the neighborhood. The optimization process starts from the target models position in the previous frame, where we rely on the smoothness property of the similarity function as mentioned, that is used for the gradient-based mean-shift approach. Color was used as the features in this project. When the tracker runs in an iteration, it is firstly assumed that the detection and localization in the initial frame is available to track (target model). Secondly a periodic analysis of each object is made in order to account for possible updates of the target models due to significant changes in color, meaning more updates (fast motion moving targets require more mean-shift iterations). The current frame target position must be within the attraction area of the similarity function (the bandwidth h of the kernel), for the hill-climbing optimization process to be successful.

$$\hat{y}_1 = \frac{\sum_{i=1}^{n_h} \mathrm{x}_i\, w_i g\left(\left\|\frac{\hat{y}_0 - \mathrm{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\hat{y}_0 - \mathrm{x}_i}{h}\right\|^2\right)}, w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_0)}}\delta(b(\mathrm{x}_i) - u)$$

Where $g(x) = -k'(x)$ is differentiable for $x \epsilon [0, \infty]$ except at a finite number of points.

Algorithm (procedure)

1. Assumptions: The target model $\{\hat{q}_u\}$ exists for all $u = 1, \ldots, m$. The tracked object location in the previous frame $\hat{y}_0$ is known.
2. Use the previous frame target location $\hat{y}_0$ as the initial location of the target candidate in the current frame, compute $\{\hat{p}_u(\hat{y}_0)\}$ for $u = 1, \ldots, m$ and compute:
$$\rho[\hat{\mathrm{p}}(\hat{y}_0), \hat{\mathrm{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{y}_0)\hat{q}_u}$$
3. Derive weights $\{w_i\}$ for $i = 1, \ldots, n_h$.
4. Determine the new location $\hat{y}_1$ of the target candidate (mean-shift step).
5. Compute the new likelihood value $\{\hat{p}_u(\hat{y}_1)\}$ for $u = 1, \ldots, m$ and evaluate:
$$\rho[\hat{\mathrm{p}}(\hat{y}_1), \hat{\mathrm{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{y}_1)\hat{q}_u}$$
6. If the similarity between the new target region and the target model is less than the one between the old target region and the target model:
$$\rho[\hat{\mathrm{p}}(\hat{y}_1), \hat{\mathrm{q}}] < \rho[\hat{\mathrm{p}}(\hat{y}_0), \hat{\mathrm{q}}]$$
perform the remaining operations of this step—move the target region half way between the new and old locations:
$$\hat{y}_1 := \tfrac{1}{2}(\hat{y}_0 + \hat{y}_1)$$
and evaluate the similarity function in this new location:
$$\rho[\hat{\mathrm{p}}(\hat{y}_1), \hat{\mathrm{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{y}_1)\hat{q}_u}$$
Return to the beginning of this step 6.
7. If $\|\hat{y}_1 - \hat{y}_0\| < \epsilon$ stop. Otherwise use the current target location as a start for the new iteration, i.e., $\hat{y}_0 := \hat{y}_1$ and continue with step 3.

Step 6 is only included to avoid potential numerical problems of the mean-shift maximization, which is a rare event. In practice this step may be omitted. Meaning that the computation of Bhattacharyya coefficient can be avoided in step 2 and 5, given that step 6 is totally avoided [1]. This additionally speeds up the tracker. This compromise has been chosen for this project for the tradeoff of efficiency.

Improvements

Dynamical background models/subtractions would improve the tracking a lot by enhancing the moving objects and reducing the background pixels or components. Background-Weighted Histogram described in [2], could be used, or a more complex Gaussian Mixture background maintenance could be used for this purpose [3]. Practical experience from the results section shows that the size of the kernel really matters, so an adaptive kernel size is preferred and is described in paper [1]. To improve the tracking further, active contour can capture the shape of the tracking object described in [4]. Last it should be mentioned that the kernel-based target localization can be integrated with a linear dynamical state space model "The Kalman Filter". This can improve the tracking even better under conditions of clutter, partial occlusion and noisy low cost surveillance cameras [5], by estimating the kernel-based target localization state with the Kalman filter.

Reference

[1] **Dorin, Comaniciu, Visvanathan Ramesh and Peter Meer. 2003.** *Kernel-Based Object Tracking.* [IEEE TRANSACTION ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE] 2003.

[2] **Ballard, M. Swain and D. 1991.** *Color Indexing.* [Int'l Conf. Computer Vision, vol. 7, no.1, pp.11-32] 1991.

[3] **L., Stauffer C. and Grimson W. E. 1999.** *Adaptive background mixture models for real-time tracking.* [CVRP '99: Computer Society Conference on Computer Vision and Pattern Recognition, Ft. Collins, USA, volume 2, pages 246-252] 1999.

[4] **Qiang Chen, Quan-Sen Sun, Phen Ann Heng, De-Shen Xia. 2010.** *Two-Stage Object Tracking Method Based on Kernel and Active Contour.* [IEEE TRANSACTION ON CIRUITS AND SYSTEMS FOR VIDEO TECHNOLOGY] 2010.

[5] **Hutchinson, K. Nickels and S. 2002.** *"Estimation Uncertainty in SSD-Based Feature Tracking".* [Image

Results

The recordings were captured by a small digital 12.2 mega pixel Samsung camera in a home-video fashion, meaning that the camera was shaking and the indoor light scene had to be enhanced (contrast adjustment) before further processing. The features that were used for the kernel based tracker, were the RGB colors which fell into place as a natural choice and was a recommendation according to the original paper [1].

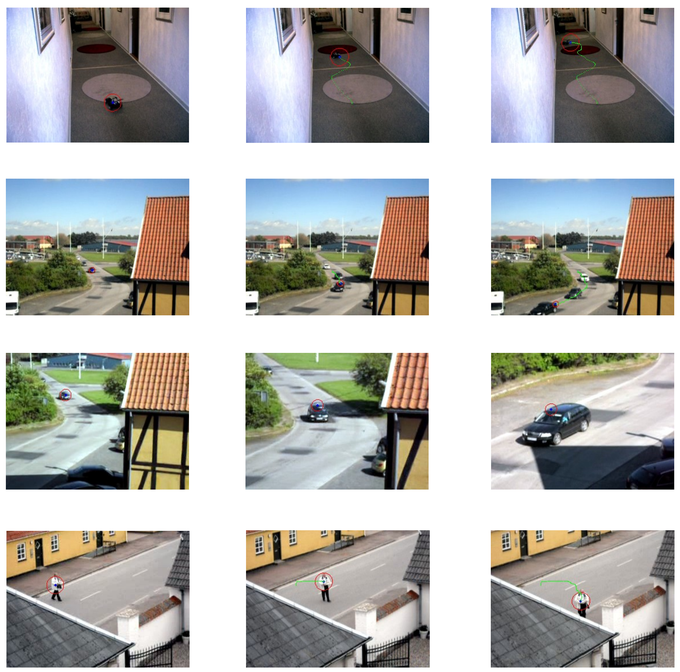

Figure 1 - (a) The first row with the following frames shows the indoor scene with the small remote car with the frames 30, 81 and 136 (left-right). (b) The second row shows the outdoor long distance scene with the frames 500, 611 and 701. (c) The third shows the outdoor zooming scene with the frames 30, 201 and 351. The fourth row shows a walking pedestrian with the frames 1, 80, and 180.

The first video recording scene which is the indoor scene in figure 1.a shows the small remote car that was recorded at a rate of 24 fps and with a kernel size of 15 pixels. An image enhancement method called "contrast stretching", was used to improve the quality of the contrast. The kernel tracker followed the small car all the way successfully.
The second recording scene in figure 1.b shows the tracking ability at a long range, only tracking a relative small target, which is a black car. It tracks the car all the way, but with a bigger kernel it would not track properly. The kernel size is 5.
The third recording scene in figure 1.c show the tracking ability in an zoom environment, where the tracker still tracks the black car though zooming is being used. The size of the kernel is 8 and a bigger kernel would give rise to problems.
The Fourth recording scene in figure 1.d show the ability to track non-rigid object which in this case is a walking pedestrian.

Conclusion

It can be concluded that the size of the kernel really has a significant improvement on the tracking precision as empirically proven, since the kernel size has been changed for various scenes as mentions in the result section. This gives rise to implement the adaptive kernel size in a future project as the first improvement. We saw from the results section that it was able to track a car from a long range. It was also able to track a car under zooming conditions. Last it was able to track a walking pedestrian, showing that the tracker was able to track a non-rigid object. Further background modeling, active contours and Kalman prediction could be used to improve the tracker significant, making it more suited for complex scenes like urban inner cities, in the environment of more partial occlusion, clutter, distractors, noise, camera motion and blurring as mentioned, but with the tradeoff of efficiency, which is necessary for real-time applications.

DTU Informatics
Department of Informatics and Mathematical Modelling