Robust Scale-Adaptive Mean-Shift for Tracking

Tomas Vojir[1], Jana Noskova[2], and Jiri Matas[1]

[1] The Center for Machine Perception, FEE CTU in Prague
Karlovo namesti 13, 121 35 Prague 2, Czech Republic
{vojirtom,matas}@cmp.felk.cvut.cz
[2] Faculty of Civil Engineering, CTU in Prague
Thakurova 7/2077, 166 29 Prague 6, Czech Republic
noskova@fsv.cvut.cz

Abstract. Mean-Shift tracking is a popular algorithm for object tracking since it is easy to implement and it is fast and robust. In this paper, we address the problem of scale adaptation of the Hellinger distance based Mean-Shift tracker.

We start from a theoretical derivation of scale estimation in the Mean-Shift framework. To make the scale estimation robust and suitable for tracking, we introduce regularization terms that counter two major problem: (i) scale expansion caused by background clutter and (ii) scale implosion on self-similar objects. To further robustify the scale estimate, it is validated by a forward-backward consistency check.

The proposed Mean-shift tracker with scale selection is compared with recent state-of-the-art algorithms on a dataset of 48 public color sequences and it achieved excellent results.

Keywords: object tracking, mean-shift, scale estimation.

1 Introduction

The Mean-Shift (MS) algorithm [4] is a non-parametric mode-seeking method for density functions. In the Computer Vision field, the Mean-Shift was introduced by Comaniciu et al. [3] who proposed its use for, inter alia, object tracking. The Mean-Shift algorithm tracks by minimizing a distance between two probability density functions (pdfs) represented by a reference and candidate histograms. Since the histogram distance (or, equivalently, similarity) does not depend on spatial structure of the search window, the method is suitable for deformable and articulated objects.

One problem the Mean-Shift algorithm suffers from is a fixed search window (i.e. a fixed object scale). When an object becomes larger, the localization becomes poor since not all pixels belonging to the object are included in the search window and the similarity function has local maxima on parts of the object. If the object become smaller, the kernel window includes background clutter which often leads to tracking failure.

Already the seminal paper [3] considered the problem and proposed to change the window size by a constant factor ($\pm 10\%$) and to run the algorithm multiple times. The scale maximizing the similarity to the target model was chosen. The approach does not cope well with increases of object size since the smaller windows usually have higher similarity and therefore the scale is often underestimated.

Collins [2] exploited image pyramids - an additional Mean-Shift procedure is used for scale selection after establishing the location. The method works well for objects with a fixed aspect ratio, which does not hold for a non-rigid or a deformable objects, therefore negating the strength of the Mean-Shift algorithm. The method is significantly slower than the standard MS.

Image moments are used in [1,10] to determine the scale and orientation of the object. The second moments are computed from an image of weights that are proportional to the probability that a pixel belongs to the target model. Yang et al. [14] introduced a new similarity measure which leads to scale estimation by comparison of second moments of target model and target candidate.

Pu et al. [12] assume a rigid object and a motion consisting of translation and scaling. The object is first tracked using the Mean-Shift both in forward and backward direction in times estimate the translation. The scale is then estimated from feature points matched by an M-estimator with outlier rejection. Similarly, [8,16] rely on ”support features” for scale estimation after the Mean-Shift algorithm solves for position. Liang et al. [8] search for the object boundary by correlating image with four templates. Positions of the boundaries directly determine the scale of the object. Zhao et a. [16] exploit affine structure to recover object relative scale from feature point correspondences between consecutive frames. Methods depending on feature matching are able to estimate the scale quite robustly, but they cannot be seamlessly integrated to the Mean-Shift framework. Moreover, estimating scale from feature correspondences takes times and has difficulties dealing with a non-rigid or a deformable object.

We present a theoretically justified scale estimation mechanism which, unlike the method listed above, relies solely on the Mean-Shift procedure for Hellinger distance. As a second contribution, we present two mechanisms which make the scale estimation more robust in the presence of background clutter and render the resulting tracker competitive with the state-of-the-art tracking methods. This is demonstrated in a comparison with the recent state-of-the-art algorithms on a large tracking dataset.

2 Mean-Shift Tracker with Scale Estimation

2.1 Standard Kernel-Based Object Tracking

In standard Mean-Shift tracking [3], the target is modelled as an m-bin kernel-estimated histogram in some feature space located at the origin:

$$\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots m} \quad \sum_{u=1}^{m} \hat{q}_u = 1. \tag{1}$$

A target candidate at location $\mathbf{y}$ in the subsequent frame is defined by its histogram

$$\hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1\ldots m} \quad \sum_{u=1}^{m} \hat{p}_u = 1; \tag{2}$$

Let $\mathbf{x}_i$ denote pixel locations, n be the number of pixels of the target model and let $\{\mathbf{x}_i^*\}_{i=1\ldots n}$ be the pixel locations of the target model centered at the origin. Spatially, the target extends over a unit circle and an isotropic, convex and monotonically decreasing kernel profile $k(x)$, is used. Function $b : R^2 \to 1 \ldots m$ maps the value of the pixel at location $\mathbf{x}_i$ to the index $b(\mathbf{x}_i)$ of the corresponding bin in the feature space. The probability of the feature $u \in \{1, \ldots, m\}$ in the target model is computed as follows:

$$\hat{q}_u = C \sum_{i=1}^{n} k\left(\|\mathbf{x}_i^*\|^2\right) \delta[b(\mathbf{x}_i^*) - u], \tag{3}$$

where δ is the Kronecker delta and C is a normalization constant so that $\sum_{u=1}^{m} \hat{q}_u = 1$.

Let $\{\mathbf{x}_i\}_{i=1 \ldots n_h}$ be pixel locations in the current frame where the target candidate is centered at location $\mathbf{y}$ and n_h is a number of pixels of the target candidate. Using the same kernel profile $k(x)$, but with a scale parameter h , the probability of the feature $u = 1 \ldots m$ in the target candidate is

$$\hat{p}_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right) \delta[b(\mathbf{x}_i) - u] \tag{4}$$

where C_h is a normalization constant. The difference between probability distribution $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1 \ldots m}$ and $\{\hat{p}_u(\mathbf{y})\}_{u=1 \ldots m}$ is measured by Hellinger distance of probability measures, which is known to be a metric:

$$H(\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}) = \sqrt{1 - \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]}\ , \tag{5}$$

where

$$\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y}) \hat{q}_u} \tag{6}$$

is the Bhattacharyya coefficient between $\hat{\mathbf{q}}$ and $\hat{\mathbf{p}}(\mathbf{y})$. Minimizing the Hellinger distance is equivalent to maximizing the Bhattacharyya coefficient $\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]$. The search for the new target location in the current frame starts at the location $\hat{\mathbf{y}}_0$ of the target in the previous frame using gradient ascent with a step size equivalent to the mean-shift method. The kernel is repeatedly moved from current location $\hat{\mathbf{y}}_0$ to the new location

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i w_i g\left(\left\|\frac{(\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{(\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)}, \tag{7}$$

where

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \delta[b(\mathbf{x}_i) - u] \tag{8}$$

and $g(x) = -k'(x)$ is the derivative of $k(x)$, which is assumed to exist for all $x \geq 0$, except for a finite set of points.

2.2 Scale Estimation

Let us assume that the scale changes frame to frame in an isotropic manner[1]. Let $\mathbf{y} = (y^1, y^2)^T, \mathbf{x}_i = (x_i^1, x_i^2)^T$ denote pixel locations and N be the number of pixels of the image. A target is represented by an ellipsoidal region $\frac{(x_i^{*1})^2}{a^2} + \frac{(x_i^{*2})^2}{b^2} < 1$ in the image and an isotropic kernel with profile $k(x)$ as in [3], restricted by a condition $k(x) = 0$ for $x \geq 1$, is used. The probability of the feature $u \in \{1, .., m\}$ in the target model is then computed as

$$\hat{q}_u = C \sum_{i=1}^{N} k\left(\frac{(x_i^{*1})^2}{a^2} + \frac{(x_i^{*2})^2}{b^2}\right) \delta[b(\mathbf{x}_i^*) - u], \tag{9}$$

where C is a normalization constant. Let $\{\mathbf{x}_i\}_{i=1\ldots N}$ be the pixel locations of the current frame in which the target candidate is centered at location $\mathbf{y}$. Using the same kernel profile $k(x)$ the probability of the feature $u = 1 \ldots m$ in the target candidate is given by

$$\hat{p}_u(\mathbf{y}, h) = C_h \sum_{i=1}^{N} k\left(\frac{(y^1 - x_i^1)^2}{a^2 h^2} + \frac{(y^2 - x_i^2)^2}{b^2 h^2}\right) \delta[b(\mathbf{x}_i) - u], \tag{10}$$

where

$$C_h = \frac{1}{\sum_{i=1}^{N} k\left(\frac{(y^1 - x_i^1)^2}{a^2 h^2} + \frac{(y^2 - x_i^2)^2}{b^2 h^2}\right)}. \tag{11}$$

The parameter h defines the scale of the target candidate, and thus the number of pixels with non-zero values of the kernel function.

For a given kernel and variable h, C_h can be approximated in the following way. Let n_1 be the number of pixels in the ellipsoidal region of the target model and let n_h be the number of pixels in the ellipsoidal region of the target candidate with a scale h; then $n_h \doteq h^2 n_1$. Using the definition of Riemann integral we obtain:

$$\sum_{i=1}^{N} k\left(\frac{(x_i^1)^2}{a^2 h^2} + \frac{(x_i^2)^2}{b^2 h^2}\right) \frac{\pi a b h^2}{n_h} \approx$$

$$\approx \int\int_{\left\{\frac{(x^1)^2}{a^2h^2} + \frac{(x^2)^2}{b^2h^2} < 1\right\}} k\left(\frac{(x^1)^2}{a^2 h^2} + \frac{(x^2)^2}{b^2 h^2}\right) dx^1 dx^2 = h^2 ab \int\int_{\|\mathbf{x}\|<1} k(\|\mathbf{x}\|^2). \tag{12}$$

Therefore $C_h \approx C \frac{1}{h^2}$ and for any two values h_0, h_1 $C_{h_1} \approx C_{h_0} \frac{h_0^2}{h_1^2}$.

As in [3] the difference between probability distribution $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots m}$ and $\{\hat{p}_u(\mathbf{y}, h)\}_{u=1\ldots m}$ will be measured by the Hellinger distance. Using the approximations above for C_h in some neighbourhood of h_0 we get

$$\rho[\hat{\mathbf{p}}(\mathbf{y}, h), \hat{\mathbf{q}}] \approx \hat{\rho}(\mathbf{y}, h) = \sum_{u=1}^{m} \sqrt{C_{h_0} \frac{h_0^2}{h^2} \sum_{i=1}^{N} k\left(\frac{(y^1 - x_i^1)^2}{a^2 h^2} + \frac{(y^2 - x_i^2)^2}{b^2 h^2}\right) \delta[b(\mathbf{x}_i) - u] \hat{q}_u} \tag{13}$$

[1] Generalization to the anisotropic where $\mathbf{h} = (h^1, h^2)^T$ is straightforward.

Thus to minimize the Hellinger distance, function $\hat{\rho}(\mathbf{y}, h)$ has to be maximized using a gradient method. In the proposed procedure the kernel with a scale parameter h_0 is iteratively moved from current location $\hat{\mathbf{y}}_0$ in direction of $\bigtriangledown\hat{\rho}(\hat{y}_0^1, \hat{y}_0^2, h_0)$ to the new location $\hat{\mathbf{y}}_1$, changing its scale to h_1.The basic idea of this procedure is the same as the idea of mean-shift method.

Let us denote

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0, h_0)}} \delta[b(\mathbf{x}_i) - u], \tag{14}$$

$$G = \sum_{i=1}^{N} w_i g\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right), \tag{15}$$

and

$$\mathbf{m}_k(\hat{\mathbf{y}}_0, h_0) = \frac{\sum_{i=1}^{N} \mathbf{x}_i w_i g\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G} - \hat{\mathbf{y}}_0, \tag{16}$$

where $\mathbf{m}_k(\hat{\mathbf{y}}_0, h_0) = \left(m_k^1(\hat{\mathbf{y}}_0, h_0), m_k^2(\hat{\mathbf{y}}_0, h_0)\right)^T$. Then we get

$$\frac{\partial\hat{\rho}(\mathbf{y}, h)}{\partial y^1}(\hat{\mathbf{y}}_0, h_0) = \frac{C_{h_0}}{a^2 (h_0)^2} \cdot G \cdot m_k^1(\hat{\mathbf{y}}_0, h_0), \tag{17}$$

$$\frac{\partial\hat{\rho}(\mathbf{y}, h)}{\partial y^2}(\hat{\mathbf{y}}_0, h_0) = \frac{C_{h_0}}{b^2 (h_0)^2} \cdot G \cdot m_k^2(\hat{\mathbf{y}}_0, h_0) \tag{18}$$

and

$$\frac{\partial\hat{\rho}(\mathbf{y},h)}{\partial h}(\hat{\mathbf{y}}_0, h_0) = \frac{C_{h_0}}{(h_0)^2} \cdot G \cdot \left[\frac{1}{h_0}\frac{\sum_{i=1}^{N} w_i \cdot \left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2}\right) \cdot g\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G} \right.$$
$$\left. - h_0 \frac{\sum_{i=1}^{N} w_i \cdot k\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G}\right]. \tag{19}$$

Finally, we obtain the mean-shift update of $\mathbf{y}$ ad h:

$$\hat{y}_1^1 = \frac{1}{a^2} m_k^1(\hat{\mathbf{y}}_0, h_0) + \hat{y}_0^1, \quad \hat{y}_1^2 = \frac{1}{b^2} m_k^2(\hat{\mathbf{y}}_0, h_0) + \hat{y}_0^2 \tag{20}$$

$$h_1 = \left[1 - \frac{\sum_{i=1}^{N} w_i \cdot k\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G}\right] h_0$$
$$+ \frac{1}{h_0} \frac{\sum_{i=1}^{N} w_i \cdot \left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2}\right) \cdot g\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G}. \tag{21}$$

3 The Tracking Algorithm

For target representation, we employ the widely used background weighting proposed in [9]. The target model $\hat{\mathbf{q}}$ is multiplied (bin-wise) by the background histogram $\hat{\mathbf{r}}$ extracted from the neighborhood of the target in the first frame and transformed such that zero bins $\hat{\mathbf{r}}_u = 0$ are set to 1 and non-zero bins $\hat{\mathbf{r}}_u \neq 0$ are set to minimum value over all bins divided by current bin value $\min_u \hat{\mathbf{r}}_\mathbf{u} / \hat{\mathbf{r}}_\mathbf{u}$. For more details see [9].

During the implementation of scale estimation we noticed a difference in the MS behaviour when the position and when the scale estimation is imprecise. While errors in position are usually corrected later on during the Mean-Shift iteration, the error in scale estimation has no "self correcting" ability in the presence of non-trivial background. The second issue with the adaptive scale is the scale ambiguity of a self-similar objects. The scale ambiguity usually leads to decrease of scale and tracking failure.

To cope with this observation and make the tracking more robust, we proposed a Mean-Shift algorithm with regularized scale estimation (MS$_s$). The algorithm is summarized in Alg. 1.

Algorithm 1. MS$_s$ – Mean-Shift with regularize scale estimation.

```
Input: Target model $\hat{\mathbf{q}}$, starting position $\mathbf{y}_0$ and starting object size $\mathbf{s}_0$
Output: Position $\mathbf{y}_t$ and scale $h_t$
$t = 1$;
repeat
    Compute $\hat{p}_u(\mathbf{y}_{t-1}, h_{t-1})$ using Eq. 10 and weights $w_i$ by Eq. 14;
    Update position $\mathbf{y}_t$ according to Eq. 20 neglecting the constants $a, b$;
    Update scale $h_t$ according to Eq. 21 + $\left[ rs(h_{t-1}) + rb(\mathbf{y}_{t-1}, h_{t-1}) \right]$;
    $t = t + 1$;
until $\left( \| \mathbf{y}_t - \mathbf{y}_{t-1} \|^2 < \varepsilon \ \mathit{AND} \ \ | h_t - h_{t-1} | < \xi \right) \ \mathit{OR} \ \ t > \mathit{maxIter}$;
```

The structure of the algorithm is similar to the standard Mean-Shift algorithm, except the scale update step. Two term regularization is introduced in the scale update step. The first term rs reflects our a priori assumption that the target scale does not change drastically, therefore we penalize the change of scale according to Eq. 22.

$$rs(h) = \begin{cases} -\log(h) & |\log(h)| \leq b_2 \\ b_2 & \log(h) < -b_2 \\ -b_2 & \log(h) > b_2 \end{cases} \tag{22}$$

where the h is scaling factor and the function is bounded by interval $(-b_2, b_2)$.

The second term rb address the problem of the scale ambiguity by forcing the search window to include a portion of background pixels. In other words, from the possible range of scales (generated by the object self-similarity) bias towards the largest. The rb function is define by Eq. 23.

$$rb(\mathbf{y}, h) = \begin{cases} \varrho - B(\mathbf{y}, h) & |\varrho - B(\mathbf{y}, h)| \leq b_1 \\ -b_1 & \varrho - B(\mathbf{y}, h) < -b_1 \\ b_1 & \varrho - B(\mathbf{y}, h) > b_1 \end{cases} \tag{23}$$

where $(\mathbf{y}, h)$ are the position and scaling factor and ϱ define the percentage of weighted background pixels that should be contained in the search window. The function is also bounded by the interval $(-b_1, b_1)$. The percentage of weighted background pixels is computed as follows:

$$B(\mathbf{y}, h) = \sum_{i=1}^{n} \delta[\hat{q}_{b(\mathbf{x}_i)}] \sum_{u=1}^{m} \hat{p}_u \delta[b(\mathbf{x}_i) - u] \bigg/ \sum_{i=1}^{n} \sum_{u=1}^{m} \hat{q}_u \delta[b(\mathbf{x}_i) - u] \tag{24}$$

where the numerator is the sum of bin weights of target candidate for pixels in which the target model has $\hat{\mathbf{q}}_u = 0$ and the divisor is the sum of bin weights of the target model over all pixels.

The MS$_s$ algorithm works well for sequences with scale, but for sequences without scale or with a significant background clutter, the algorithm tends to estimate non-zero scale, which may lead to continuous wrong scale estimation and tracking failure. Therefore, we present a technique to overcome the difficult sequence parts by so called *Backward scale consistency check*. The Backward check use reverse tracking from position $\mathbf{y}_t$ obtained by forward tracking and validating the estimated scaling factors from step $t-1$ to t and t to $t-1$. This validation ensures that in the presence of background clutter the scale estimation does not "explode" and enables tracker to recover. The algorithm using this technique is summarize in Alg. 2.

Algorithm 2. MS$_{fb}$ – Mean-Shift with scale and backward consistency check.

```
Input: Target model $\hat{\mathbf{q}}$, starting position $\mathbf{y}_0$ and starting object size $\mathbf{s}_0$
Output: Position and scale in each frame $(\mathbf{y}_t, \mathbf{s}_t)$, where $t \in \{1, \ldots, n\}$
foreach Frame $t \in \{1, \ldots, n\}$ do
    $[\mathbf{y}_t, h]$ = MS$_s(q$, image$_t$, $\mathbf{y}_{t-1}, \mathbf{s}_{t-1})$;
    if $|log(h)| > \Theta_s$ then
        // Scale change - proceed with consistency check
        $[\sim, h_{back}]$ = MS$_s(q$, image$_{t-1}$, $\mathbf{y}_t, h\mathbf{s}_{t-1})$;
        if $|log(h * h_{back})| > \Theta_c$ then
            // Inconsistent scales
            $\mathbf{s}_t = (1 - \alpha - \beta)\mathbf{s}_{t-1} + \alpha\mathbf{s}_{\text{default}} + \beta h\mathbf{s}_{t-1}$   where   $\alpha = c_1(\frac{\mathbf{s}_{\text{default}}}{\mathbf{s}_{t-1}})$;
    else
        $\mathbf{s}_t = (1 - \gamma)\mathbf{s}_{t-1} + \gamma h\mathbf{s}_{t-1}$;
```

In case of detected scale inconsistency the object size is a weighted combination of three parts: (i) previous size; (ii) new estimated size; (iii) "default" size, which in our case is initial size of the object. This combination is hand design to reflect our experimental observation and requirements for the scale adaptability of the MS$_s$ and for stability of the standard Mean-Shift algorithm.

We also notice that Mean-Shift is more stable if the bandwidth size is biased toward a larger size, therefore the computation of the weight α (from Alg. 2) is not symmetric but prioritize enlarging the object size. In our case, the default size is constant during tracking, and preliminary experiments with adapting the default size shows no significant benefits and only introduced error caused by wrong updates.

4 Experimental Protocol

Experiments were conducted on 48 color sequences[2] collected from published literature. The sequences vary in length from dozens of frames to thousands, contain diverse object types (rigid, articulated), have different scene settings (indoor/outdoor, static/moving camera, lightning conditions). Object occlusions and disappearance from the field of view are also present in the data.

The proposed Mean-Shift algorithms (MS$_s$ and MS$_{fb}$) are compared with the standard published Mean-Shift algorithm (MS) and its scale adaptation (MS$_\pm$) proposed in [3]. Both standard algorithms carried out the background weighting [9], so that the underlying target weighting is the same.

The proposed methods are also compared with the state-of-the-art tracking algorithms that are available as source code, namely SOAMST [11] base on Mean-Shift algorithm, LGT [13], TLD [6], CT [15] and STRUCK [5]. Parameters for these algorithms were left default as set by the authors. Note that our results may differ from other publications, since we did not optimize their parameters for the best performance for each sequence (e.g. as was done in [15]), but were fixed for all experiments. Moreover, the target was initialized in the first frame using the ground truth position for all algorithms. Since the main part of the CT and STRUCK algorithms are randomized, we run them 10 times on each sequence and take the average result.

Performance of the algorithms was assess by the recall (the number of correctly tracked frames divided by number of frames where the target is visible). Recall was chosen because some algorithm exhibit detector-like behavior, therefore other frequently used criteria such as first failure frame or failure from which the algorithm did not recover will not capture the real performance of the algorithm, i.e. in how many frames the algorithm locate the target correctly. The frame was consider tracked correctly if the overlap with the ground truth was higher than 0.5 measured as $o = \frac{area(T \cap G)}{area(T \cup G)}$, where *T* is object bounding box reported by the tracker and *G* is ground truth bounding box.

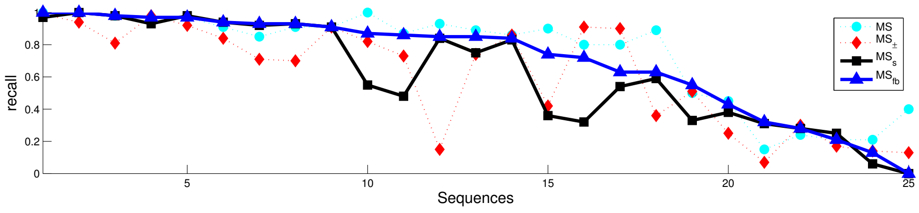
We measure also the speed of the algorithms as the average processing time per frame. Note that the algorithms are not implemented in the same programming language (SOAMST, LGT, TLD, CT using matlab with MEX files, STRUCT and Mean-Shift using C++) which may bias the speed measurement towards the more efficient programming language.

The Mean-Shift algorithms are written in the C++ without heavy optimization and without multithreading. All parameters of the algorithm were fixed for all experiments. Some of the parameters are fairly standard (Mean-Shift termination criterion) and the rest were chosen empirically as follows: bounds for regularization terms $b_1 = 0.05$, $b_2 = 0.1$ and $\varrho = 0.2$; termination of the Mean-Shift algorithm $\varepsilon = 0.1$, $\xi = 0.01$ and $maxIter = 15$; scale consistency check $\Theta_s = 0.05 \approx 5\%$ of the scale change, $\Theta_c = 0.1$; exponential averaging $c_1 = 0.1$, $\beta = 0.1$ and $\gamma = 0.3$. The pdf is represented as a histogram computed over the RGB space and quantized into the $16 \times 16 \times 16$ bins.

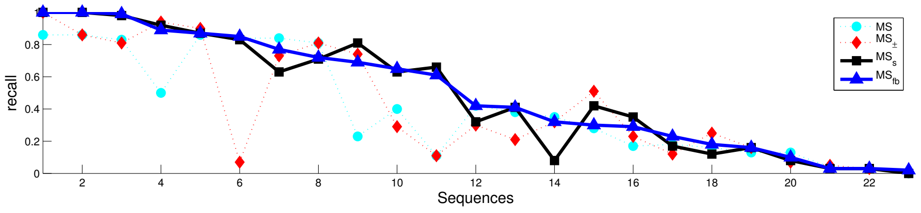
[2] `http://cmp.felk.cvut.cz/~vojirtom/dataset`

5 Results

Since many of the sequences has small or no scale change, the MS algorithm scores better than MS$_\pm$ and MS$_s$ in the average recall. However, by validating the scale change the MS$_{fb}$ tracker is always better (or equal) than the individual algorithm MS or MS$_s$ and outperforming both of them in total average recall. The result are shown in the left side of the Table 1 for sequences that contain object scale and visualize in the Fig. 1b and in Table 2 (visualization in Fig. 1a) for sequences without the scale.



(a) Sequence without significant scaling of the object.



(b) Sequence with scaling of the object.

Fig. 1. Comparison of the proposed MS$_{fb}$ (MS$_s$) and standard Mean-Shift algorithms on sequences without (a) and with (b) significant scaling. Sequences (x-axis) are sorted by the recall measure of the MS$_{fb}$ algorithm.

Comparison of the MS$_{fb}$ and state-of-the-art algorithms is illustrated in Fig. 2, which shows that on majority of the sequences the performance of the MS$_{fb}$ is higher than other, but also show that MS$_{fb}$ is not suitable for some sequences (e.g. *CarChase, Motocross, Volkswagen*). These "long-term" sequences containing object disappearance from the field of view, scene cuts and significant object occlusion with background clutter, and because the MS$_{fb}$ does not provide any re-detection ability it can not handle these cases. In these sequences the TLD tracker achieving the best results.

The highest score achieving the MS$_s$ and MS$_{fb}$ on the *Vid_?* dataset [7], since the sequences contain small amount of background clutter and out-of-plane/in-plane rotation, which is difficult to deal for the state-of-the-art algorithm, where the representation of the object is usually spatial dependent and out/in-plane rotation is not explicitly modeled.

Performance of the Mean-Shift algorithms, in general, drops when there is a significant background clutter. This issue is more significant for the case where the tracker estimating more parameters (such as translation and scale) and therefore the estimation

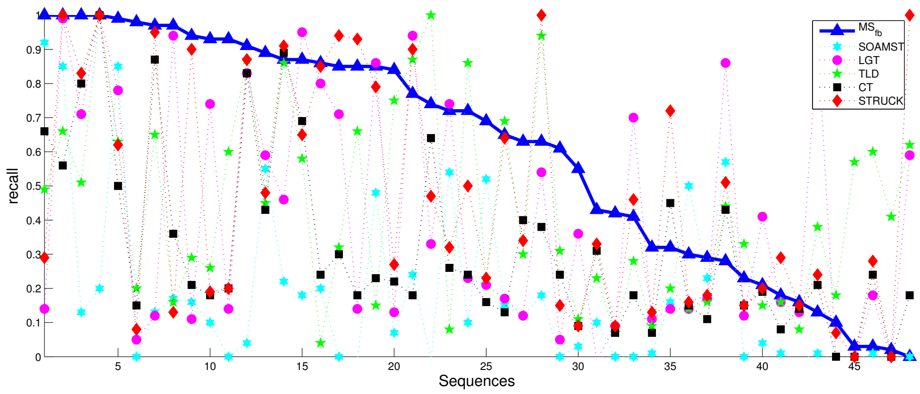


Fig. 2. A Comparison of the MS$_{fb}$ and the state-of-the-art algorithms on all sequences. Sequences (x-axis) are sorted by the recall measure of the MS$_{fb}$ algorithm.

Table 1. Recall on sequences with scale change. Bold text - best result for the sequence, underscore - second best. na indicates the program crashes.

Sequence \ Method	MS$_s$	MS$_{fb}$	MS	MS$\pm$	SOAMST	LGT	TLD	CT	STRUCK
girl	0.63	$\underline{0.65}$	0.40	0.29	0.15	0.17	**0.69**	0.13	0.64
surfer	**0.35**	$\underline{0.29}$	0.17	0.23	0.23	0.17	0.16	0.11	0.18
Vid_A	**1.00**	**1.00**	0.86	**1.00**	$\underline{0.92}$	0.14	0.49	0.66	0.29
Vid_C	$\underline{0.92}$	0.89	0.50	**0.94**	0.55	0.59	0.45	0.43	0.48
Vid_E	0.87	0.87	0.86	$\underline{0.90}$	0.22	0.46	0.86	0.89	**0.91**
Vid_G	**1.00**	**1.00**	$\underline{0.86}$	$\underline{0.86}$	0.13	0.71	0.51	0.80	0.83
Vid_K	0.63	0.77	0.84	0.73	0.24	**0.94**	0.87	0.18	$\underline{0.90}$
Vid_L	**0.81**	0.69	0.23	$\underline{0.74}$	0.52	0.21	0.23	0.16	0.23
gymnastics	0.42	0.30	0.28	**0.51**	$\underline{0.50}$	0.14	0.14	0.15	0.16
cliff-dive2	**0.16**	**0.16**	0.13	**0.16**	na	0.13	0.08	0.14	$\underline{0.15}$
motocross1	0.12	0.18	0.16	$\underline{0.25}$	0.01	0.16	0.16	0.08	**0.29**
skiing	$\underline{0.32}$	**0.42**	**0.42**	0.30	0.00	0.09	0.07	0.07	0.09
CarChase	0.08	0.10	$\underline{0.13}$	0.07	na	na	**0.18**	0.00	0.07
Motocross	0.00	$\underline{0.02}$	0.00	0.01	na	na	**0.41**	0.00	0.00
Panda	0.17	$\underline{0.23}$	0.18	0.12	0.00	0.12	**0.33**	0.15	0.15
Volkswagen	0.03	0.03	0.04	$\underline{0.05}$	na	na	**0.57**	0.00	0.00
pedestrian3	**0.66**	$\underline{0.61}$	0.11	0.11	0.00	0.05	0.31	0.24	0.15
jump	0.08	$\underline{0.32}$	**0.35**	$\underline{0.32}$	0.01	0.11	0.09	0.07	0.13
Asada	0.71	0.72	**0.81**	**0.81**	0.54	$\underline{0.74}$	0.08	0.26	0.32
drunk2	0.03	0.03	0.03	0.03	0.01	0.18	**0.60**	0.24	$\underline{0.28}$
woman	0.83	$\underline{0.85}$	0.84	0.07	na	0.14	0.66	0.18	**0.93**
lemming	$\underline{0.98}$	**0.99**	0.83	0.81	0.85	0.78	0.63	0.50	0.62
trellis	0.41	0.41	0.38	0.21	0.00	**0.70**	0.28	0.18	$\underline{0.46}$
Mean	0.49	0.50	0.41	0.41	0.27	0.34	0.38	0.24	0.36

corruption induces more vital drift (in scale dimension) than in the case of estimating a pure translation. This was mainly the case of sequence *drunk2, car11 and dinosaur* where the color distribution of the target was similar to a background.

Table 2. Recall on sequences without a scale change. Bold text - best result for the sequence, underscore - second best. na indicates the program crashes.

Method Sequence	MS$_s$	MS$_{fb}$	MS	MS$\pm$	SOAMST	LGT	TLD	CT	STRUCK
Vid_B	0.97	**1.00**	**1.00**	0.99	0.85	0.99	0.66	0.56	**1.00**
Vid_D	0.93	0.97	0.96	**0.98**	0.13	0.12	0.65	0.87	0.95
Vid_F	0.38	0.43	**0.45**	0.25	0.10	na	0.23	0.31	0.33
Vid_H	**1.00**	**1.00**	**1.00**	0.94	0.20	**1.00**	**1.00**	**1.00**	**1.00**
Vid_I	0.83	0.84	**0.86**	**0.86**	0.07	0.13	0.75	0.22	0.27
Vid_J	0.54	0.63	0.80	**0.90**	0.12	0.12	0.30	0.40	0.34
dinosaur	0.06	0.13	0.21	0.14	0.01	**0.93**	0.38	0.21	0.24
hand	0.92	**0.93**	0.85	0.71	0.10	0.74	0.26	0.18	0.19
hand2	0.33	**0.55**	0.50	0.51	0.03	0.36	0.11	0.09	0.09
torus	**0.98**	0.97	0.97	0.92	0.17	0.94	0.16	0.36	0.13
cliff-dive1	0.55	0.87	**1.00**	0.82	0.18	0.95	0.58	0.69	0.65
motocross2	**0.91**	**0.91**	**0.91**	**0.91**	0.04	0.83	0.87	0.83	0.87
mountain-bike	0.84	0.85	0.93	0.15	0.00	0.71	0.32	0.30	**0.94**
pedestrian4	**0.93**	**0.93**	0.91	0.70	0.00	0.14	0.60	0.20	0.20
pedestrian5	0.36	0.74	0.90	0.42	na	0.33	**1.00**	0.64	0.47
diving	0.25	0.21	0.21	0.17	0.04	**0.41**	0.15	0.19	0.20
gym	**0.94**	**0.94**	0.91	0.84	0.16	0.11	0.29	0.21	0.90
trans	0.28	0.28	0.24	0.30	0.57	**0.86**	0.44	0.43	0.51
faceocc1	0.59	0.63	0.89	0.36	0.18	0.54	0.94	0.38	**1.00**
figure_skating	0.48	0.86	**0.87**	0.73	0.20	0.80	0.04	0.24	0.85
board	0.75	0.85	**0.89**	0.74	0.48	0.86	0.15	0.23	0.79
box	0.31	0.32	0.15	0.07	0.16	0.14	0.20	0.45	**0.72**
liquor	0.32	0.72	0.80	**0.91**	0.10	0.23	0.86	0.24	0.50
car11	0.00	0.00	0.40	0.13	0.00	0.59	0.62	0.18	**1.00**
person	**0.98**	**0.98**	**0.98**	0.81	0.00	0.05	0.20	0.15	0.08
Mean	0.62	0.70	0.74	0.61	0.16	0.54	0.47	0.38	0.57

The validation step proof to be a beneficial for the MS$_s$, but it comes with the price of slowing the tracking by factor of two (at maximum). From the experiments (see Table 3), the slow down factor against MS$_s$ is 1.6 in average. Compared to other algorithms, both MS$_s$ and MS$_{fb}$ algorithms are slower then the MS, since the speed depends on the window size which is fixed in the case of MS, but multiple times faster then the state-of-the-art algorithms.

Table 3. Processing speed in a milliseconds. Max (min) are computed as an max (min) of the average time per sequence; mean as the average time over all sequences.

Method	MS$_s$	MS$_{fb}$	MS	MS$\pm$	SOAMST	LGT	TLD	CT	STRUCK
max	17	28	16	125	6107	864	152	36	112
min	1	1	1	3	207	107	6	11	43
mean	5	8	3	14	816	250	51	21	82

6 Conclusion

In this work, a theoretically justified scale estimation for the Mean-Shift algorithm using Hellinger distance has been proposed. The new scale estimation procedure is regularized to make it more robust.

In the cases were the scale of the object is constant throughout the video sequence, the MS$_s$ is at a disadvantage w.r.t. standard MS due to the extra degree of freedom in its motion model estimation. Therefore, we introduce a scheme (fb) for an automatic decision to either accept the newly estimated scale or to use a more robust weighted combination which is likely to reduce erroneous scale updates.

The newly proposed MS$_{fb}$ has been compared with the state-of-the-art algorithms on a very large dataset of tracking sequences and it outperforms them in average recall, and in the processing speed.

Acknowledgements. The research was supported by Technology Agency of the Czech Republic grant TE01020415 (V3C – Visual Computing Competence Center) and by the Czech Science Foundation Project GACR P103/12/G084.

References

1. Bradski, G.R.: Computer Vision Face Tracking For Use in a Perceptual User Interface. Intel Technology Journal Q2 (1998)
2. Collins, R.T.: Mean-shift blob tracking through scale space. In: Computer Vision and Pattern Recognition, pp. 234–240. IEEE Computer Society (2003)
3. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift, pp. 142–149 (2000)
4. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. Information Theory 21(1), 32–40 (1975)
5. Hare, S., Saffari, A., Torr, P.: Struck: Structured output tracking with kernels. In: International Conference Computer Vision, pp. 263–270 (November 2011)
6. Kalal, Z., Matas, J., Mikolajczyk, K.: P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. In: Conference on Computer Vision and Pattern Recognition (2010)
7. Klein, D.A., Schulz, D., Frintrop, S., Cremers, A.B.: Adaptive real-time video-tracking for arbitrary objects. In: Intelligent Robots and Systems, pp. 772–777 (October 2010)
8. Liang, D., Huang, Q., Jiang, S., Yao, H., Gao, W.: Mean-shift blob tracking with adaptive feature selection and scale adaptation. In: International Conference Image Processing (2007)
9. Ning, J., Zhang, L., Zhang, D., Wu, C.: Robust mean-shift tracking with corrected background-weighted histogram. Computer Vision, IET 6(1), 62–69 (2012)
10. Ning, J., Zhang, L., Zhang, D., Wu, C.: Scale and orientation adaptive mean shift tracking. Computer Vision, IET 6(1), 52–61 (2012)
11. Ning, J., Zhang, L., Zhang, D., Wu, C.: Scale and orientation adaptive mean shift tracking. Computer Vision, IET 6(1), 52–61 (2012)
12. Pu, J.-X., Peng, N.-S.: Adaptive kernel based tracking using mean-shift. In: Campilho, A., Kamel, M.S. (eds.) ICIAR 2006. LNCS, vol. 4141, pp. 394–403. Springer, Heidelberg (2006)
13. Čehovin, L., Kristan, M., Leonardis, A.: An adaptive coupled-layer visual model for robust visual tracking. In: 13th International Conference on Computer Vision (November 2011)
14. Yang, C., Duraiswami, R., Davis, L.: Efficient mean-shift tracking via a new similarity measure. In: Computer Vision and Pattern Recognition, vol. 1, pp. 176–183 (June 2005)
15. Zhang, K., Zhang, L., Yang, M.-H.: Real-time compressive tracking. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 864–877. Springer, Heidelberg (2012)
16. Zhao, C., Knight, A., Reid, I.: Target tracking using mean-shift and affine structure. In: ICPR, pp. 1–5 (2008)