IET Computer Vision

Research Article

IET Journals
The Institution of Engineering and Technology

Invariant foreground occupation ratio for scale adaptive mean shift tracking

ISSN 1751-9632
Received on 12th June 2014
Accepted on 24th November 2014
doi: 10.1049/iet-cvi.2014.0150
www.ietdl.org

Yi Song[1], Shuxiao Li[1], Chengfei Zhu[1] ✉, Sheng Jiang[2], Hongxing Chang[1]

[1]*The Integrated Information System Research Center, Institute of Automation Chinese Academy of Sciences, 95 Zhongguancun East Road, Beijing, People's Republic of China*
[2]*New Technology Lab, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun Institute of Optics, Changchun, People's Republic of China*
✉ *E-mail: chengfei.zhu@ia.ac.cn*

Abstract: The mean shift algorithm has been introduced successfully into the field of computer vision to be an efficient approach for visual tracking but the tracker has been awkward in handling the scale change of the object. This study addresses the scale estimation problem of the mean shift tracker, and proposes a novel method which is based on invariant foreground occupation ratio to solve this problem. The foreground occupation ratio is defined as the proportion of the foreground pixels in an image region. By taking an analysis of the foreground occupation ratio, the authors obtain its three simple properties. With its property of scale invariance, an iterative approximation approach is employed to estimate the scale of the foreground in the current image. The scale value is modified by a weighting function, and it is adjusted along the two axes with respect to the width and the height of the target. The scale estimation algorithm is then employed in the mean shift tracker to obtain the ability of scale adaptation for tracking. Experimental results show that, using the authors method for object scale estimation, the mean shift tracker performs well in tracking the target efficiently when its scale continuously changes.

1 Introduction

Real-time object tracking is a fundamental but a very important task in many computer vision applications such as video surveillance [1, 2], human–computer interaction [3, 4], robotics [5] and driver assistance [6]. Algorithms for visual object tracking are designed to tackle the difficulties existing in real applications such as occlusions, illumination variation and changes in the foreground or background [7]. In recent years, various tracking approaches have been proposed to handle these difficulties and strive for a better performance in tracking [8].

In the domain of real-time visual object tracking, the mean shift tracker [9] has achieved great success in the past decade for its good efficiency and robustness. Given the object in the previous frame, the mean shift algorithm is applied by locally shifting the kernel window to seek the most similar candidate in the current frame, with an iterative procedure. However, a drawback of the algorithm for tracking is that it uses fixed kernel window to do the shifting, resulting in the lack of its scale adaptability.

To handle this problem, one of the main approaches is to search in the scale space for the best scale of the target. In the original tracker proposed in [9], three different sizes of window are used for the shifting procedure, and the scale of the target is calculated based on the size of the window by which it obtains the most similar candidate. Collins [10] generates three-dimensional (3D) scale space using the difference-of-Gaussian spatial filters, and exploits the mean shift algorithm through the 3D space to obtain both the spatial position and the scale of the target. The scale searching approach may find a good scale for the target if sufficient scale values are searched. It brings about a large computational burden. On the contrary, if a small number of the scale values are searched (for example, three), the approach could not give a satisfying scale estimation.

Another approach is to calculate the covariance matrix of the distribution of the target pixel samples for scale estimation. Zivkovic and Krose [11] treat the mean shift algorithm as an expectation-maximisation (EM)-like algorithm, and use Gaussian kernels to approximate the distribution of the target pixels. By iteratively calculating the covariant matrix within the Gaussian kernel, the location and the scale of the target are estimated. In contrast, Ning *et al.* [12] combine the moment features within the shift window to obtain the covariance matrix of the samples, and make an adjustment to the matrix to estimate the scale of the target. These methods obtain a good estimation of the target scale if the target pixel samples are well distributed and few noise samples exist within the candidate region. If a distracter, such as a similar object, comes near the real target, then the approach may obtain a wrong result for the target scale.

The third approach estimates the scale by optimising the scale parameter in mean shift algorithm. Jiang *et al.* [13] substitute the kernel bandwidth into the density estimator. By maximising the lower bound of a likelihood function they deduced, the bandwidth matrix that gives the scale value of the target is obtained. Recently Vojir *et al.* [14] propose a gradient method to simultaneously optimise the scale parameter as well as the location in the shifting procedure, and the target scale is updated in tracking. Other than the scale optimisation, these methods consider many extra constraints to restrict the obtained scale value, which might not yield the best scale estimation for the target.

In this paper, we propose a novel method based on invariant foreground occupation ratio to solve the problem of the scale estimation of the object in tracking. For an image region containing both foreground pixels and background pixels, its foreground occupation ratio is defined as the proportion of the foreground pixels (i.e. the pixels of the target) in this region, as Fig. 1*b* shows. We take an analysis of the foreground occupation ratio, and acquire its three properties. On the basis of the scale invariance of the foreground occupation ratio, an iteration approximation algorithm is developed to estimate the scale of the target. The scale value is modified by a weighting function to be more accurate, and then it is adjusted along the two axes with respect to the width and the height of the target. Finally, the scale estimation algorithm is embedded into the mean shift procedure to obtain a scale adaptive tracker. In our method, the blob image (i.e. the sample weight image), as shown in Fig. 1*c*, is used to calculate the foreground occupation ratio. Two main

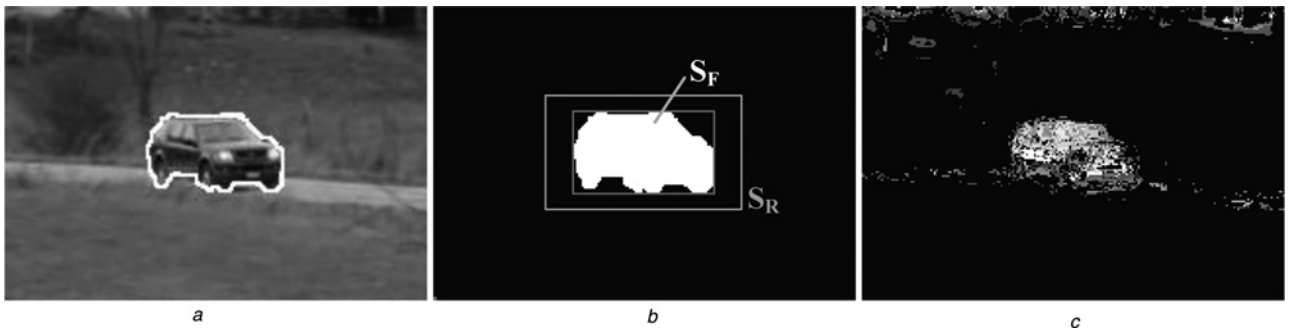

Fig. 1 *Simple illustration of the foreground occupation ratio*

a Original image
Contour of the foreground is highlighted with a white curve
b Foreground region and the background region of the original image
The target (inner) rectangle and the ROI (outer) rectangle are shown. If the area of the foreground region is S_F and the area of the ROI is S_R, then the foreground occupation ratio of the ROI can be calculated by $r_f = S_F/S_R$
c Generated blob image of the original image
Some pixels in the background are falsely labelled as foreground pixels

improvements for scale adaptive tracking are included in our method. One is that the foreground occupation ratio is exploited to estimate the scale of the object. Since the scale space calculation is avoided and replaced with direct scale calculation, our method is more efficient and accurate to estimate the scale compared with the first approach. The other improvement is that the scale value is modified using a weighting function and further adjusted. Thus, our method is more robust and it has a better scale adaptability compared with the second and the third approaches.

The remaining contents of this paper are organised as follows. Section 2 reviews some works about the blob image for mean shift tracking methods. In Section 3, an analysis of the foreground occupation ratio is taken, and the scale estimation method for the target is described in details. Section 4 depicts the scale adaptive mean shift algorithm using the proposed scale estimation method. Experimental results and the evaluation of the proposed tracker are given in Section 5. Conclusions are made in Section 6.

2 Blob image for mean shift tracking

As for visual object tracking, the target to be tracked is usually bounded by a rectangle or an ellipse in the first frame, and we use the rectangle in our study. In the original mean shift tracker [9], the appearance model of the target is represented by a kernel weighted histogram $\hat{\boldsymbol{q}} = \{\hat{q}_u\}_{u=1,\ldots,m}$ with m bins

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

where $\{\boldsymbol{x}_i^*\}_{i=1,\ldots,n}$ denotes the normalised positions of the pixels inside the target rectangle, $b(\boldsymbol{x}_i^*)$ denotes the index of its bin in the histogram and δ is the Kronecker delta function. Similarly, the target candidate histogram $\hat{\boldsymbol{p}}(\boldsymbol{y}) = \{\hat{p}_u(\boldsymbol{y})\}_{u=1,\ldots,m}$ is calculated within the candidate rectangle centred at $\boldsymbol{y}$

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

where $k(\cdot)$ denotes the spatial kernel in the equations above. The mean shift algorithm is developed to locate the most similar candidate to the target by iteratively shifting from the old position $\hat{\boldsymbol{y}}_0$ to a new location $\hat{\boldsymbol{y}}$

$$\hat{\boldsymbol{y}} = \frac{\sum_{i=1}^{n_h}\boldsymbol{x}_i w_i g\left(\left\|(\hat{\boldsymbol{y}}_0 - \boldsymbol{x}_i)/\boldsymbol{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|(\hat{\boldsymbol{y}}_0 - \boldsymbol{x}_i)/\boldsymbol{h}\right\|^2\right)} \tag{3}$$

with the weight value

$$w_i = \sum_{u=1}^{m}\sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\boldsymbol{y}}_0)}}\delta[b(\boldsymbol{x}_i) - u] \tag{4}$$

and the function $g(\cdot) = -k'(\cdot)$. The iteration process ends when the distance between $\hat{\boldsymbol{y}}_0$ and $\hat{\boldsymbol{y}}$ is tiny.

From another perspective, the mean shift locating procedure is employed in the sample weight image to seek the centroid of the blob, see (3). Each pixel in the candidate window at $\boldsymbol{x}_i$ is a weighted sample point, with its weight value w_i. From (4), we find that the sample weight value indicates the likelihood that the pixel belongs to the target. That is to say, the pixel which belongs to the target has a higher weight value. The sample weight image presents the blob image for the mean shift tracking.

The blob image could be utilised to help estimate the scale of the target in mean shift tracking. Since the foreground blob itself represents the target, the target scale can be estimated by obtaining the scale of the foreground blob. Collins [10] generates blob images in scale space, and employs the mean shift algorithm through these images to obtain the scale of the foreground blob. Bradski [15] and Ning *et al.* [12] calculate the moments of the foreground data distribution in the blob image. With the moments, they obtain the area and the eigenvalues of the distribution, by which the scale can be estimated. In [16], Liang *et al.* apply four correlation templates to the blob image, and find the locally maximal response to acquire the four boundaries of the target. In our method, the blob image is processed to separate the foreground pixel from the background, by thresholding the weight value. The pixel is labelled as the foreground pixel where the weight value is above the threshold, otherwise it is labelled as the background pixel. Then it is used to calculate the foreground occupation ratio, which is further used to estimate the target scale.

Various approaches can be taken to produce the blob image, such as using (4) or using a specific colour histogram for backprojection [15]. A more typical approach is introduced in [17], where Collins *et al.* calculate the object histogram in the target window, as well as the background histogram around the target. Then the log value of the object/background histogram for each bin is computed to generate the weight value for each colour. In this way, colours distinctive for the target have positive weight values, and those distinctive for the background have negative ones. The blob image is obtained by backprojecting the log likelihood values into the original image. It should be noted that weight values computed by (4) at the same position are changed in each iteration of the shift procedure, thus the blob image is changed in each iteration. In contrast, for our tracking method, a fixed blob image for representing the current image is needed to obtain the correct

IET Comput. Vis., 2015, Vol. 9, Iss. 4, pp. 489–499

foreground occupation ratio of a local image region. With the background information in the previous frame, we employ the method in [17] to produce the fixed blob image for the current frame. The details are described in Section 4.

3 Foreground occupation ratio for scale determination

The foreground scale estimation problem in blob images could be briefly described as follows. In the reference blob image, the foreground (i.e. the target) is bounded by a rectangle, called the target rectangle. The scale of the foreground can be measured by the size of the target rectangle, which is presented by a vector $\boldsymbol{h} = (h_x, h_y)^{\mathrm{T}}$. In the current blob image, assuming that the location of the foreground remains but the scale of the foreground is changed, the goal is to estimate the scale of the foreground in the current image.

In this section, the definition and the properties of the foreground occupation ratio are firstly introduced, and then the foreground scale estimation algorithm based on the foreground occupation ratio is described in details.

3.1 Definition and properties of the foreground occupation ratio

As shown in Fig. 1*b*, for an image region $\boldsymbol{R}$ which contains both foreground region $\boldsymbol{F}$ and background region $\boldsymbol{B}$, the foreground occupation ratio of the region is defined as $r_{\mathrm{f}} = S_{\mathrm{F}}/S_R$, where S_R denotes the area of the whole region, and S_{F} denotes that of the foreground region within. Practically, S_R can be calculated as the total number of pixels in the whole image region, and S_{F} can be calculated as the number of the foreground pixels within the image region. Therefore, the foreground occupation ratio r_{f} is calculated by

$$r_{\mathrm{f}} = \frac{\sum_{i=1}^{N} \delta_{\mathrm{f}}(\boldsymbol{x}_i)}{N} \tag{5}$$

where N denotes the number of pixels in the whole region and δ_{f} indicates whether the pixel $\boldsymbol{x}_i$ belongs to the foreground

$$\delta_f(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \in \boldsymbol{F} \\ 0, & \boldsymbol{x} \in \boldsymbol{B} \end{cases} \tag{6}$$

The foreground occupation ratio has two obvious properties with respect to the change of the foreground, which are the translational invariance and the rotational invariance. The translational invariance means that, when the foreground moves inside the image region, the foreground occupation ratio stays invariant, as in Fig. 2*a*. The rotational invariance means that, when the foreground takes an in-plane rotation, the foreground occupation ratio stays invariant, as in Fig. 2*b*. These two properties are found on the condition that the whole foreground region lies within the image region after it translates or rotates. However, the ratio do not possess the property of the scale invariance directly, since the ratio value will change when the scale of the foreground region changes.

Consider using a rectangle to bound the foreground, and keep the size ratio between the rectangle and the image region rectangle a constant when the bounding rectangle changes scale. It means that the image region for calculating the foreground occupation ratio changes with the foreground rectangle, by the same factor of the scaling. In this case, as long as the ratio of the foreground area to the area of its bounding rectangle remains invariant, the foreground occupation ratio stays the same. Typically, the property stands when the foreground isotropically changes scale, as shown in Fig. 2*c*. Therefore, the property of the scale invariance is obtained.

3.2 Scale estimation for the foreground

The scale invariance property of the foreground occupation ratio could be utilised to estimate the scale of the foreground in the current image. Assuming the target changes scale isotropically, then the scaling factor can be represented by a scalar s. In the reference image, the size of the foreground is denoted by $\boldsymbol{h}_p = (h_x^p, h_y^p)^{\mathrm{T}}$. We bound an enlarged rectangle with the same centre of the target rectangle outside the target, as the region of interest (ROI) rectangle. Then, the reference foreground occupation ratio of the ROI can be calculated as r_0. In the current image, we resize the ROI rectangle with the variable s, and calculate the foreground occupation ratio of the ROI, denoted as $r_{\mathrm{f}}(s)$. It is easy to know that $r_{\mathrm{f}}(s)$ generally decreases as s increases. According to the scale invariance property, when the ratio value $r_{\mathrm{f}}(s)$ equals to r_0, the corresponding scaling factor s indicates the scaling of the foreground between the two images.

A simple experiment is performed to test the method. In the reference blob image, the target is given by the target rectangle, and an enlarged rectangle is manually assigned as the ROI rectangle for calculating the foreground occupation ratio r_0, as shown in Fig. 3*a*. In the current blob image, as the ROI rectangle changes scale by the factor s along each axis, each value of the foreground occupation ratio is recorded as $r_{\mathrm{f}}(s)$, and drawn as the function of s, as shown in Fig. 3*c*. Let $r_{\mathrm{f}}(s)$ equals to r_0, the corresponding factor s_r is obtained, as the scaling factor of the foreground between the two images. As shown in Fig. 3*b*, when $r_{\mathrm{f}}(s)$ equals to r_0, we obtain a satisfying scaling value s_r for the foreground in the current blob image.

To acquire the scale of the foreground efficiently, we propose to use an iterative approximation approach. Consider an ideal situation as shown in Fig. 2*c*. In the reference blob image, assuming the foreground area within the ROI is denoted by S_{F}, and the area of the ROI is denoted by S_R, then we obtain $r_0 = S_{\mathrm{F}}/S_R$. In the current image, the foreground area changes into $S_{\mathrm{F}'} = s_r^2 S_{\mathrm{F}}$, and the foreground occupation ratio of the same ROI becomes $r_1 = S_{\mathrm{F}'}/S_{\mathrm{R}}$. The objective is to obtain the new ROI with the area $S_{\mathrm{R}'} = s_r^2 S_{\mathrm{R}}$ to have the foreground occupation ratio r_0,

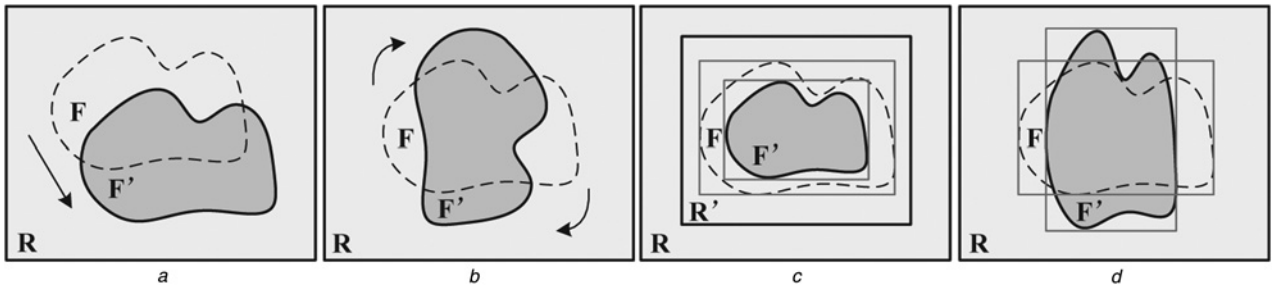

Fig. 2 *Properties of the foreground occupation ratio*

a Translational invariance, $S_{\mathrm{F}}/S_{\mathrm{R}} = S_{\mathrm{F}'}/S_{\mathrm{R}}$
b Rotational invariance, $S_{\mathrm{F}}/S_{\mathrm{R}} = S_{\mathrm{F}'}/S_{\mathrm{R}}$
c Scale invariance, $S_{\mathrm{F}}/S_{\mathrm{R}} = S_{\mathrm{F}'}/S_{\mathrm{R}'}$
d If $S_{\mathrm{F}}/S_{\mathrm{R}} = S_{\mathrm{F}'}/S_{\mathrm{R}}$, then $S_{\mathrm{F}} = S_{\mathrm{F}'}$

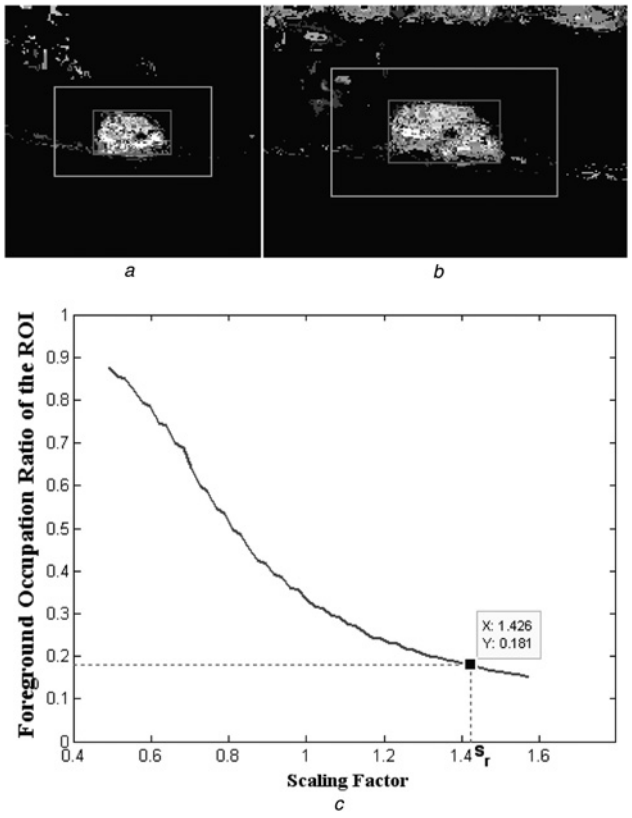

Fig. 3 *Test for foreground scale estimation*

a Target in the reference blob image, $r_0 = 0.1787$
b Estimation results of the foreground in the current blob image, using the scaling factor s_r obtained in (*c*)
c Function value $r_f(s)$ as the size of ROI changes by the scaling factor s in the current blob image. With $r_f(s) \doteq r_0$, a satisfying estimation $s_r = 1.426$ is obtained

satisfying $r_0 = S_{F'}/S_{R'}$. Then it is easy to obtain that the area of the new ROI could be calculated as $S_{R'} = (S_{F'}/r_0) = (r_1/r_0)S_R$. Therefore, the scaling of the foreground s_r can be calculated as

$$s_r^2 = \frac{r_1}{r_0} \Rightarrow s_r = \sqrt{\frac{r_1}{r_0}} \tag{7}$$

This equation means that the scaling value s_r can be determined by two foreground occupation ratio values r_0 and r_1. However, in real blob images, when the ROI rectangle is updated for r_0, the area of the foreground pixels within the new ROI may not remain as the previous value $S_{F'}$. It may increase as the ROI is enlarged and may decrease otherwise. Thus, the new ROI is smaller (when $s_r > 1$) or bigger (when $s_r < 1$) than the one which has the foreground occupation ratio r_0. Hence, to obtain the ROI that has the ratio r_0, we recalculate the foreground occupation ratio of the new ROI, denoted by r_2, and apply (7) again using r_2 and r_0 to update the new ROI. This step is iteratively taken until the ROI remains unchanged, and the scaling s_r is obtained.

If substituted $r_1 = S_{F'}/S_R$ and $r_0 = S_F/S_R$ into (7), one could find the scaling value s_r is actually associated with the ratio of the two area values of the foregrounds S_F and $S_{F'}$. It means that, for the ideal situation, the iterative approximation approach is equivalent to directly using the area ratio of the foreground within their respective ROI rectangles in the two images to estimate the scaling value. However, in real blob images, our approach is more stable than directly using the area ratio, since it always takes the foreground pixels in the background region into consideration when the ROI is updated. Furthermore, if the scaling value is large enough for the target to exceed out of the range of the same ROI rectangle in the reference image, our approach still works well but the other one will fail.

Scale modification: In real generated blob images, some pixels in the target may be represented as background pixels, and some background pixels may be falsely labelled as foreground pixels, as can be seen by comparing Figs. 1*b* and *c*. Affected by the falsely labelled pixels, the scale estimation approach presented above gives a rough scale for the target. In some extreme cases, large numbers of background pixels are falsely labelled as foreground pixels or the target pixels falsely labelled as the background pixels in the current blob image, thus the iterative approximation approach might converge to a much larger (when $s_r > 1$) or a much smaller (when $s_r < 1$) scaling value than the real one.

A weighting approach is developed in our method to reduce the impact of the falsely labelled pixels for a better estimation of the scale. Let s_r denote the scaling value obtained by the iterative approximation approach. The rate of the foreground area change between the ROI in the reference image and the one estimated in the current image could be computed as

$$\Delta r_c = |s_r^2 - 1| \tag{8}$$

If the scale estimation approach is seriously affected by the falsely labelled pixels, the value of Δr_c intend to be abnormally large (i.e. very close to 1 when $s_r < 1$ or much larger than 1 when $s_r > 1$). For obtaining a reasonable scale, a weighting function with respect to Δr_c should be developed to indicate if the estimated scaling value is reliable. Numerous options exist for the function, and we pick up the sigmoid-like function in our method

$$w(\Delta r_c) = 1 - \frac{1}{1 + e^{a(\Delta r_c - b)}} \tag{9}$$

where a and b are two constants that determine the characters of the function. The constant b indicates the maximum rate of the area change that is reliable for the scaling, and the constant a indicates the declining rate of the weight value around b. With the weighting function, the scaling value is modified to acquire a more reasonable one

$$s_c = \sqrt{1 + w(\Delta r_c)(s_r^2 - 1)} \tag{10}$$

To obtain an effective weighting function, the parameter a in (9) is set to give a steep declining rate around b. The value chosen for parameter b is important, since it determines the upper bound of the scale adaptability of the tracker. A smaller b declines the scale adaptability of the tracker, whereas a larger b declines the effectiveness of the scale modification. In practice, b should be set as the supposed maximum area change rate of the foreground. For visual object tracking, two consecutive frames are captured in quite a short time (for example, 40 ms). Thus, in most cases the scale of the target is smoothly changed between the reference image (i.e. the previous image) and the current image. Therefore, the parameter b in the weight function is set to be 0.2 indicating that the change rate of the foreground area should be <20%, and the parameter a is set to be 50. This function works well in most of tracking sequences in our experiments. As shown in Fig. 4*c*, affected by the falsely labelled pixels in the current blob image, the scaling value estimated by the iterative approximation approach becomes inappropriate for the foreground. After it is modified by the weighting function, the impact of the falsely labelled pixels is reduced and a better scale estimation for the foreground is obtained, as shown in Fig. 4*d*.

3.3 Foreground scale adjustment

Above, the foreground is assumed to change scale isotropically, and the scaling value s_c is acquired. Mostly, the scale of the foreground changes in an anisotropic manner along the two axes with respect to the width and the height of the target. Since the same foreground occupation ratio is used, the area of the obtained foreground region within the ROI should be the same as the one that is

IET Comput. Vis., 2015, Vol. 9, Iss. 4, pp. 489–499

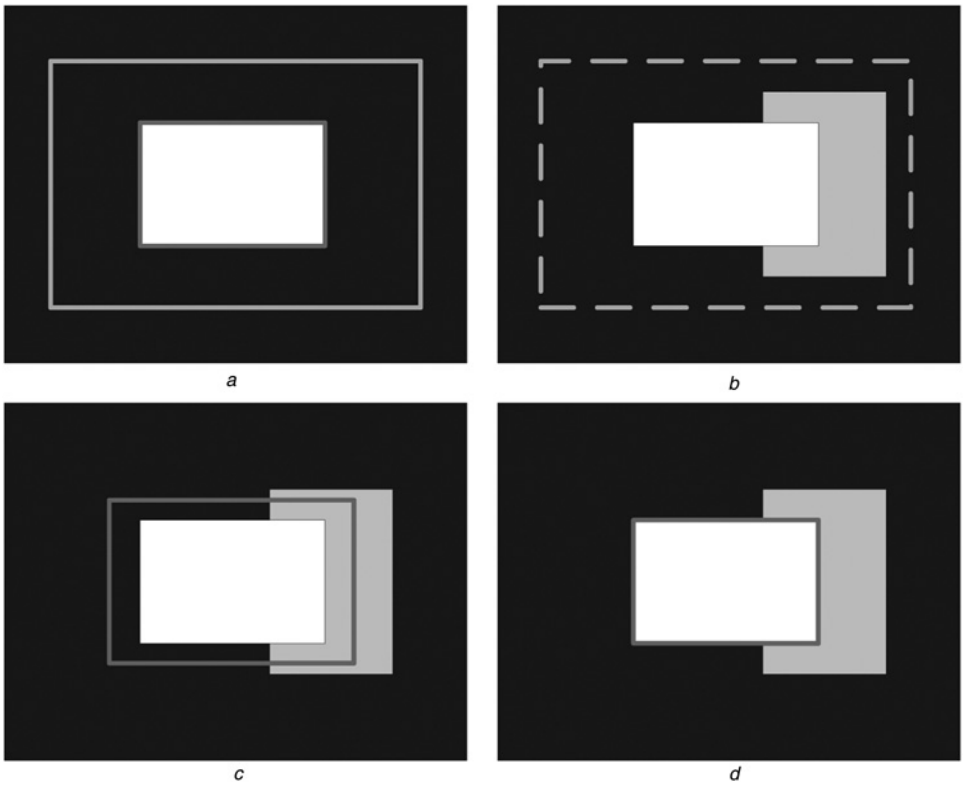

Fig. 4 *Illustration for scale modification*

a Foreground (in white, bounded by the inner rectangle) and the ROI (bounded by the outer rectangle) in the reference blob image
b Foreground pixels in the current blob image, within the same ROI in (*a*)
Pixels in grey are background pixels but they are falsely labelled as the foreground pixels, the area of which equals 75% of the real foreground region
c Target rectangle estimated by the approximation approach
d Target rectangle obtained after the scale modification procedure
This scale is a better one than that in (*c*) to bound the foreground

isotropically changed, as shown in Fig. 2*d*. On the basis of this inference, we make an adjustment to the width and the height of the candidate rectangle to obtain the final scale of the foreground.

This procedure is implemented as follows. First, we generate N different rectangles centred at the same location of the candidate rectangle. These rectangles have different sizes $\boldsymbol{h}_c^i = \left(s_x^i h_x^p,\ s_y^i h_y^p\right)^{\mathrm{T}},\quad i = 1,\ 2,\ \ldots,\ N$, yet with the same area as the candidate, which means $s_x^i s_y^i = s_c^2$. Then, the one that contains most foreground pixels is picked up as the final estimated rectangle of the target, with the size $\boldsymbol{h}_c = \left(s_x h_x^p,\ s_y h_y^p\right)^{\mathrm{T}}$. A larger N might yield a more accurate estimate but with a heavier computational burden. In our implementation, N is chosen to be five to obtain satisfying results. One is the candidate rectangle, and the widths of the other four rectangles are adjusted by ±5%, ±10% with respect to the width of the candidate rectangle. The procedure of the scale adjustment could be efficiently realised using the integral image [18].

When the number of the falsely labelled pixels in the background is large, the scale adjustment procedure might be seriously affected and obtain a wrong size of the foreground. Therefore, the procedure should be restricted by a condition with respect to the number of the pixels labelled as foreground in the background window. This condition may be dependent on the size of the ROI rectangle. In our algorithm, the area of the ROI rectangle is three times larger than that of the target rectangle. If the number of the labelled foreground pixels in the background window, denoted as S_{fb}, is less than 50% of that in the target candidate window, denoted by S_{ft}, then the procedure is employed. On the other side, if S_{fb} exceeds S_{ft}, we consider that the scale should also be adjusted. For this situation in practice, the number of the correctly labelled pixels in the background window probably exceeds the number of the falsely labelled pixels, so that the scale adjustment procedure could also be employed. Moreover, it could compensate the estimation error of the scale modification procedure in the case that the scale of the target quickly changes in the tracking.

The whole scale estimation algorithm is referred in Algorithm 1 (see Fig. 5).

4 Scale adaptive mean shift tracker

As previously mentioned, for the visual tracking, the target in the first frame is assumed to be bounded by a target rectangle. In our method, the appearance model of the target is represented by the background weighted histogram [19], $\hat{\boldsymbol{q}}_{\mathrm{b}} = \{\hat{q}_u^{\mathrm{b}}\}_{u=1,\ldots,m}$. It is calculated relating to the ratio of the normalised object histogram $\hat{\boldsymbol{h}}_{\mathrm{ob}}$ and the normalised background histogram $\hat{\boldsymbol{h}}_{\mathrm{bk}}$ for each bin, where $\hat{\boldsymbol{h}}_{\mathrm{ob}}$ is computed within the target window, and $\hat{\boldsymbol{h}}_{\mathrm{bk}}$ is computed in a background window around the target window.

The blob image is generated using a likelihood ratio technique [17]. The log likelihood ratio indicates the probability of a pixel for which it belongs to the foreground, computed by

$$L_{\mathrm{f}}(u) = \log \frac{\max(\hat{q}_u^{\mathrm{b}},\ \varepsilon)}{\max(\hat{h}_{\mathrm{bk}}(u),\ \varepsilon)} \tag{11}$$

With the log likelihood ratio, the value for each pixel in the blob image can be given as

$$\boldsymbol{I}_{\mathrm{b}}(\boldsymbol{x}) = \begin{cases} L_{\mathrm{f}}(b(\boldsymbol{x})), & L_{\mathrm{f}}(b(\boldsymbol{x})) > L_\theta \\ 0, & L_{\mathrm{f}}(b(\boldsymbol{x})) \le L_\theta \end{cases} \tag{12}$$

Algorithm 1.

Input: Foreground occupation ratio r_0, initial target size $\mathbf{h}_p$

Output: Current target size $\mathbf{h}_c$

1. Compute the foreground occupation ratio of the same ROI in the reference image, denoted as r_1;
2. Initialise $s_r = 1$;
3. **while** $(r_1 \neq r_0)$
   Calculate s_{rt} using equation (7);
   Update the ROI using s_{rt}, compute the foreground occupation ratio r_2;
   $s_r \leftarrow s_r \cdot s_{rt}$, $r_1 \leftarrow r_2$;
   end while
4. Obtain the modified scaling value s_c using s_r with equation (8)(10);
5. Compute S_{fb} and S_{ft};
6. **if** $(0.5S_{ft} < S_{fb} < S_{ft})$
   $\mathbf{h}_c = s_c\mathbf{h}_p = (s_c h_x^p, s_c h_y^p)^T$;
   else
   Adjust the candidate size $s_c\mathbf{h}_p$ to obtain $\mathbf{h}_c = (s_x h_x^p, s_y h_y^p)^T$;
   end if

Fig. 5 *Scale estimation based on foreground occupation ratio*

where the function $b(\boldsymbol{x})$ gives the corresponding histogram bin for the pixel at position $\boldsymbol{x}$ and L_θ is a threshold with positive value indicating if this pixel is labelled as foreground. In our method, the threshold is set to be $L_\theta = \lambda L_{\max}$, where $L_{\max}$ denotes the maximum value of $L_f(u)$. Hence, in the blob image, pixels which have positive values are labelled as foreground, and those with zero are labelled as background. Note that the background histogram $\hat{\boldsymbol{h}}_{\text{bk}}$ has been used twice to generate the blob image, which makes it more discriminative for separating the foreground from the background, thus more robust to employ the proposed scale estimation algorithm.

The mean shift algorithm is employed to locate the target, where the Gaussian kernel is used as the spatial kernel $k(\cdot)$. The algorithm is exploited in the same way as the original tracker [9] using (3) and (4) for the locating, independent from the blob image generated by (12). Actually, the mean shift procedure could be exploited in the blob image to locate the target. However, it is not taken in our method. The reason is that, in the blob image, the pixel value outstands where the colour feature is distinctive from the background. The shifting procedure applied in the blob image will converge to the distinctive part of the target rather than the centre of it, which makes the tracker easier to lose the target. The blob image is utilised to calculate the foreground occupation ratio only, for estimating the scale of the target. Actually, the scale estimation algorithm could be easily employed into other trackers for their scale adaptability, since the scale estimation procedure and the target locating procedure are relatively independent. Our method embeds the method into a mean shift tracker since the tracker is the most efficient one among state-of-the-art trackers [8].

After the mean shift locating converges, the scale estimation algorithm is employed. Then the obtained scaling value is used to update the shift window for a new round of locating. The procedure is iteratively taken until the converged location obtained by the mean shift locating of two rounds remains unchanged.

After localisation, the log likelihood ratio is recalculated using the target model and the background window in the current frame, and the blob image is recomputed for updating the foreground occupation ratio r_0. This value is then applied in the next frame for the scale estimation. The whole tracking algorithm is summarised in Algorithm 2 (see Fig. 6).

5 Experiments and discussion

In this section, the experiments on establishing and evaluating our proposed tracker are given in detail. The proposed tracker can be named as IFORMS, which is short for the invariant foreground occupation ratio based mean shift tracker. In the implementation of the IFORMS tracker for real-time object tracking, the target histogram is calculated in the red–green–blue colour space quantised in $16 \times 16 \times 16$ bins. The image region for calculating the target histogram should contain sufficient background pixels, so that the size of the region is chosen to be 3×3 times the size of the target window. Note that the image region for calculating the histogram is not necessarily the same with the ROI which is used for computing the foreground occupation ratio, for which we use 2×2 times the size of the target window.

5.1 Datasets and evaluation metric

The image sequences 'CarScale, Woman, Boy, Doll and Lemming' from [20] and 'jump, Vid_B, Vid_C, Vid_E and Vid_K' from [21] are used in our experiments as the evaluation datasets, where the target changes scale a lot. Additionally, some other challenges for tracking are included in these sequences, such as the background clutter (in 'Vid_K, Vid_B, Vid_C, Doll and Lemming'), the partial occlusion (in 'Vid_E, CarScale, Woman, Doll, Lemming and jump') and the illumination change (in 'Doll, Woman, Boy, jump and Vid_K').

The measurement used for evaluation is the dice coefficient metric [22], calculated as

$$D = \frac{2 \cdot \text{Area}(\Omega_X \cap \Omega_G)}{\text{Area}(\Omega_X) + \text{Area}(\Omega_G)} \tag{13}$$

IET Comput. Vis., 2015, Vol. 9, Iss. 4, pp. 489–499

Algorithm 2.

Input: Target model $\hat{\mathbf{q}}_b$, initial target location $\mathbf{y}_0$ and its size $\mathbf{h}_0$, initial foreground occupation ratio r_0, initial $L_f(u)$

Output: Target location $\mathbf{y}_t$ and its size $\mathbf{h}_t$ in each frame t

For each frame t, t=1,2,...,n, **do**

1. Apply $L_f(u)$ in frame t to obtain the current blob image by equation (12);
2. $\mathbf{y}^{(0)} \leftarrow \mathbf{y}_{t-1}$, $\mathbf{h}^{(0)} \leftarrow \mathbf{h}_{t-1}$
3. **for** m=1 to MaxIter

   Apply the mean shift algorithm using equation (3),(4) to get converged from $\mathbf{y}^{(m-1)}$ to $\mathbf{y}^{(m)}$;

   With the current blob image, update target size $\mathbf{h}^{(m)}$ by Algorithm 1 using r_0, $\mathbf{h}^{(m-1)}$;

   if $\|\mathbf{y}^{(m)} - \mathbf{y}^{(m-1)}\| < \epsilon_y$

   break;

   end for;
4. $\mathbf{y}_t \leftarrow \mathbf{y}^{(m)}$, $\mathbf{h}_t \leftarrow \mathbf{h}^{(m)}$
5. Recalculate $L_f(u)$ by equation (11) using $\hat{\mathbf{q}}_b$ and the background information. Recalculate r_0.

Fig. 6 *Scale adaptive mean shift tracking*

where Ω_X denotes the estimated target rectangle and Ω_G denotes the groundtruth rectangle. This measurement indicates the tracking accuracy by measuring the degree of the overlap between Ω_X and Ω_G, which is suitable to evaluate scale adaptive trackers.

5.2 Parameter for blob image generation

As Section 4 mentioned, the blob image is generated by (12). A blob image is robust if it can exactly separate the foreground from the background. A robust blob image to represent the original image is required for the proposed tracker, since the performance of the scale estimation algorithm is dependent on whether the foreground occupation ratio can be well computed. Thus, an experiment is designed for obtaining a proper value of λ. The basis of the experiment is that, for the same object (especially non-rigid) appearing in different backgrounds, the foreground occupation ratio of the ROI calculated using the groundtruth target rectangle should be approximately the same. Image sequences 'Vid_B, CarScale, and Boy' are used in the experiment. For each sequence, the foreground occupation ratio of the ROI in each frame is computed with respect to variable λ, using the groundtruth target rectangle. Then the mean value and the standard deviation of the ratio in the sequence are calculated as a function of λ. If the blob images are well generated, the foreground occupation ratio computed in each frame should be about the same, which is close to the target-to-ROI area ratio. Therefore, a good λ yields a small standard deviation value, meanwhile a large mean value for the ratio. As Fig. 7 shows, λ could be properly chosen in the interval [0.2, 0.3], where the minimum value of the standard deviation is obtained.

5.3 Scale adaptability discussion

To demonstrate the scale adaptability of the new tracking algorithm, an experiment is carried out by comparing our proposed tracking algorithm with an EM-like tracker (EMShift) [11] in a real generated blob image. The blob image for testing is the same image as in Fig. 3*b*. The performance of EMShift and IFORMS is shown in Figs. 8*a* and *b*. The estimated results for the foreground in each iteration are exhibited in the two subimages. For the EMShift tracker, the results are exhibited in ellipse as the algorithm requires, and the final estimation is exhibited in the bounding rectangle for comparing.

As the figure illustrates, IFORMS can give better estimation results of the foreground than EMShift, with a few iterations. The EMShift tracker performs well when the distribution of the foreground data is Gaussian, otherwise it is easily trapped into a local optimal solution, especially when the number of the outliers is large.

As mentioned in Section 3.2, parameter b determines the upper bound of the scale adaptability. A simple illustration of the effectiveness of parameter b for the scale adaptability of IFORMS is shown in Fig. 8*c*. The area change rate between the initial rectangle and the final rectangle is about 3.24, so that b could be set to 4 to indicate that the maximum area change rate is 4. As shown in Figs. 8*b* and *c*, with a larger b value, the IFORMS could handle larger scale change of the target. Note that the objective of this experiment is to clearly illustrate how the parameter b affects the scale adaptability of the tracker, and how the algorithm estimates the scale in each iteration. In normal applications of visual tracking, parameter b is set to 0.2 to balance the scale adaptability with the effectiveness of the scale modification, and it works well in most cases.

5.4 Tracking results

The IFORMS tracker is compared with various scale adaptive trackers using the ten image sequences to evaluate its performance. The trackers for the comparison are the original scale modified mean shift tracker (MS_3) [9], the scale adaptive blob tracker (BLOBT) [10], the EM-like tracker (EMShift) [11], the scale-estimated mean shift (SEMS) tracker [13], the scale and orientation adaptive mean shift tracker (SOAMST) [12] and the MS_{fb} tracker [14]. For the trackers to be compared, the parameters are set in default as their works introduced.

The tracking results are shown in Table 1. For each sequence, the dice coefficient generated in each frame is calculated, and then the

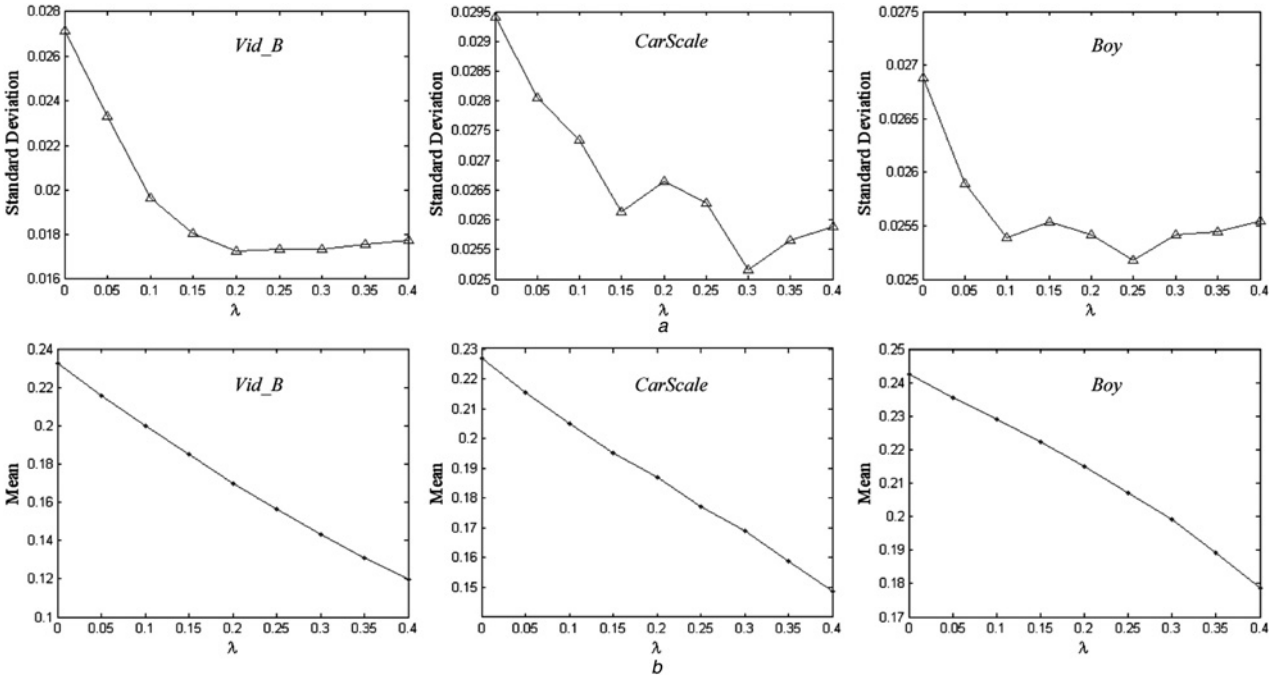

Fig. 7 *Standard deviation and the mean value of the foreground occupation ratio with respect to λ in the test sequences*

a Standard deviation
b Mean value

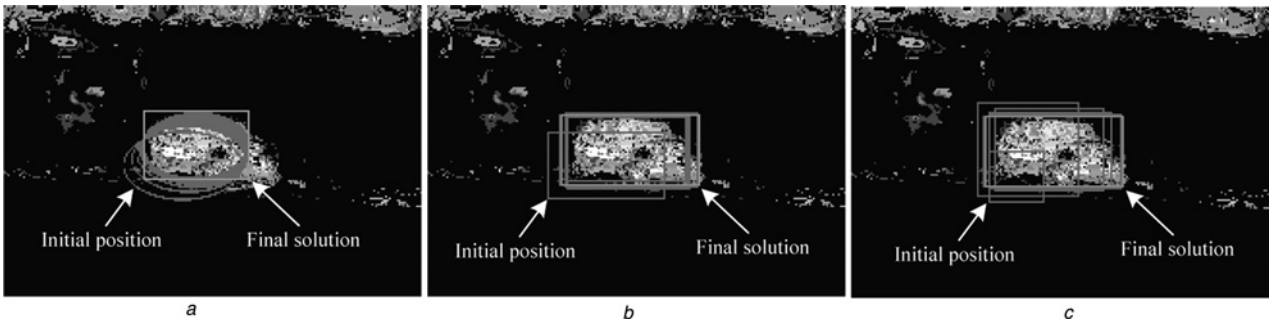

Fig. 8 *Performance of the two trackers in a real blob image*

a EMShift iterations
b IFORMS iterations with $b = 0.2$
c IFORMS iterations with $b = 4.0$
Green rectangle in each subimage shows the convergence of each tracker. Even the position and the scale of the initial rectangle is far different from the groundtruth value, IFORMS can give a satisfying result of the foreground estimation both in location and scale, with a quick convergence

Table 1 Accuracy of different trackers

Seq.	IFORMS	MS_3	BLOBT	EM Shift	SEMS	SOAMST	MS_{fb}
CarScale	**0.7972**	0.4285	0.6851	0.6296	0.4751	0.5989	0.7967
Woman	0.5736	0.3215	0.152	0.0399	0.469	0.4671	**0.691**
Boy	0.7985	**0.8307**	0.1059	0.4729	0.7665	0.8124	0.8166
Doll	0.6965	0.6685	0.3356	0.6809	0.7203	0.6356	**0.7881**
Lemming	**0.8373**	0.7522	0.2605	0.7738	0.7628	0.8127	0.8175
Jump	0.4882	0.4083	0.2777	0.4945	0.1376	**0.5866**	0.0933
Vid_B	0.8051	0.823	0.1553	**0.8306**	0.726	0.8214	0.7485
Vid_C	0.6891	**0.7438**	0.1194	0.3922	0.4202	0.5935	0.4335
Vid_E	**0.8941**	0.8043	0.6931	0.8232	0.8456	0.7057	0.8662
Vid_K	0.7577	**0.8473**	0.5191	0.7307	0.2796	0.6698	0.5691
mean	**0.7337**	0.6628	0.3304	0.5868	0.5603	0.6704	0.6621

The highest value in each row is shown in bold, and the second highest value is underlined

average value over the sequence is recorded. The IFORMS tracker and the MS_{fb} tracker perform well in most sequences. However, MS_{fb} could not work well when the size ratio of the target changes quickly, as its bad performance in the sequence jump. The average performance of our proposed tracker is more satisfying compared with other trackers.

IET Comput. Vis., 2015, Vol. 9, Iss. 4, pp. 489–499

5.4.1 Fast scale change: In the *CarScale* sequence shown in Fig. 9, the target undergoes fast scale change. BLOBT, SEMS, SOAMST and MS_{fb} can give a good scale estimation when the scale change is relatively small at the beginning, as shown in frames 62 and 108. However, when the target scale changes quickly in consecutive frames, the scale value estimated by these trackers becomes very poor, as shown in frame 232. Note that SEMS loses the target when it goes through partial occlusion. EMShift, MS_3 could not accurately estimate the target scale even when the scale change is small, partly because the target location is not well estimated. On the contrary, our proposed tracker could give accurate estimations in both scale and the location for the target, even the scale of the target changes fast.

5.4.2 Background clutter: In addition to the small scale change of the target, the background clutter is a challenge in the 'Vid_K' sequence, as shown in Fig. 10. EMShift and SOAMST are easily distracted by similar colour features in the background and give

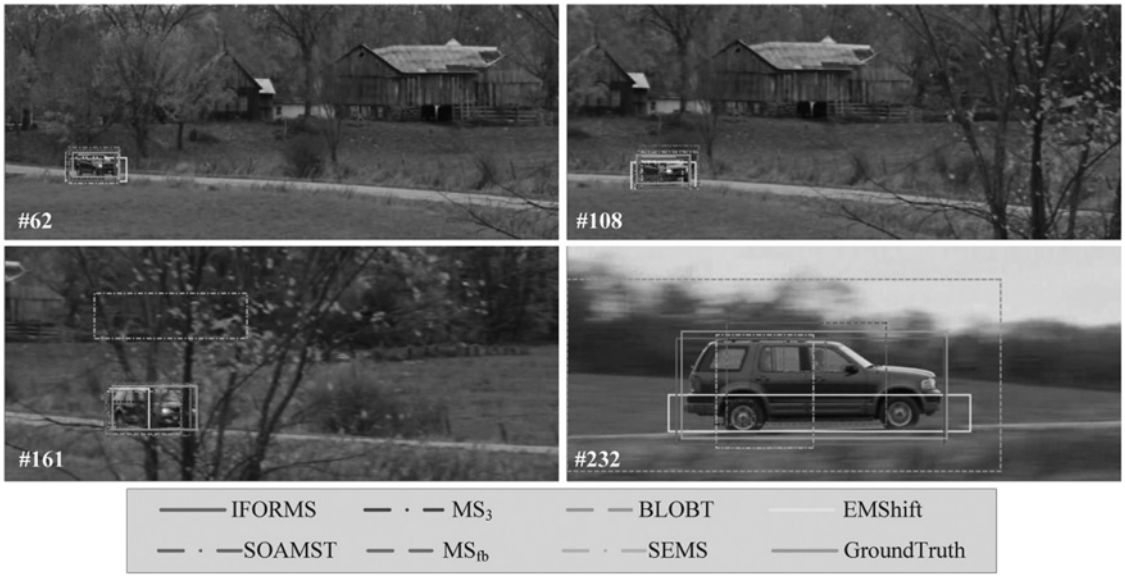

Fig. 9 *Sample images of the tracking results in CarScale*
Frame number of each image in the sequence is labelled in the bottom-left

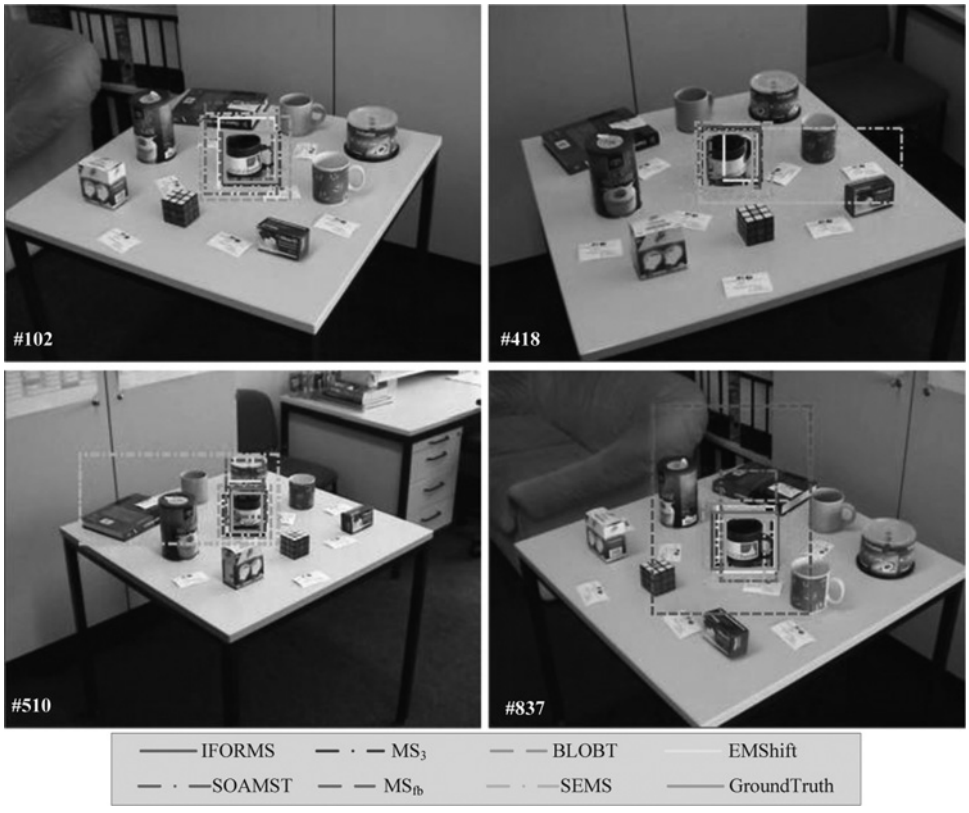

Fig. 10 *Sample images of the tracking results in Vid_K*
Frame number of each image in the sequence is labelled in the bottom-left

Fig. 11 *Sample images of the tracking results in Vid_E*
Frame number of each image in the sequence is labelled in the bottom-left

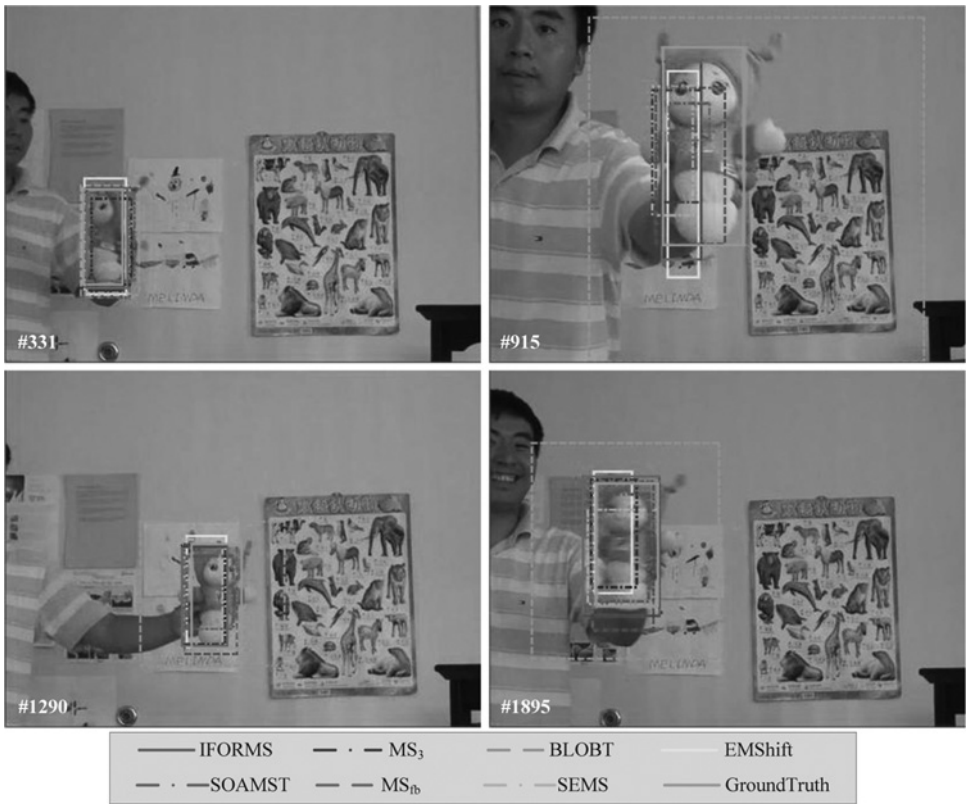

Fig. 12 *Sample images of the tracking results in Doll*
Frame number of each image in the sequence is labelled in the bottom-left

IET Comput. Vis., 2015, Vol. 9, Iss. 4, pp. 489–499

Table 2 Average time cost of different trackers

Trackers	IFORMS	MS_3	BLOBT	EMShift	SEMS	SOAMST	MS_{fb}
time per frame, ms	1.98	0.41	54.83	1.85	5.06	1.1	3.15

unsatisfying scale estimations, as shown in frames 102, 510 and 837. The reason is that the covariance matrix-based methods are easily affected by the pixel sample distribution around the target. SEMS, MS_{fb} and BLOBT may expand their bounding rectangles for the target when similar colour feature exists around the target. In comparison, our proposed tracker, as well as the MS_3 tracker, does not expand the estimated rectangles and gives good scale estimation. This benefits from the scale modification strategy in our method.

5.4.3 Partial occlusion: The scale change in the 'Vid_E' sequence is caused by partial occlusion, as shown in Fig. 11. In this case, the target width changes quickly while the target height remains the same, which means that the aspect ratio is changed during the tracking. The figure shows that SEMS gives better results than EMShift, MS_3, SOAMST, BLOBT but not as good as our tracker, which can also be seen in Table 1. MS_{fb} can give good estimations when the target can be totally observed but it loses accuracy of the scale when the aspect ratio of the target greatly changes, as shown in frame 177. In contrast, our tracker could handle the scale change by partial occlusion much better than MS_{fb}, as the scale adjustment procedure helps the tracker to obtain the accurate size for the target.

5.4.4 Illumination change: The target in the 'Doll' sequence goes through illumination change in addition to the scale change, as shown in Fig. 12. In frame 915, the target appears near the screen and becomes brighter than the target in the first frame. In frame 1290, the man turns on the light and the target reflects different colours. In both cases, our proposed tracker fails to give good estimations for the target. The reason is that the blob image for estimating the location and the scale is generated based on the initial target model. When the colour of the target changes, the foreground pixels in the blob image could not represent the target well. Like our tracker, the other trackers could not give good estimations in case one, as shown in frame 915. In case two, MS_{fb} yields a better result than our tracker, as shown in frame 1290. When the colour of the target returns similar to the target model, our tracker again gives accurate estimation, as shown in frame 1895.

5.5 Efficiency comparison

To compare the computing efficiency of all the trackers, the Vid_K sequence is used for evaluation. The sequence contains 1020 frames with the image size 320×240, and the average size of the target is about 33×35. The experiments are conducted in C++ implementation on a desktop computer with an Intel Core 2 Duo 2.66 GHz central processing unit, 2 GB random access memory. The time cost per frame for all the trackers are shown in Table 2. IFORMS performs more efficient than MS_{fb} and much more efficient than BLOBT, while MS_3 takes the lead. The good efficiency of our tracker makes it competent for real-time processing tasks.

6 Conclusions

A novel scale adaptive mean shift tracker based on invariant foreground occupation ratio is described in this paper. The foreground occupation ratio has three simple but useful properties. With its property of the scale invariance, the scale change of the target between frames could be safely estimated, thus an accurate tracking can be achieved by the proposed tracker. Experimental results show that the proposed tracker has a good performance in handling the scale change of the target, with a real-time efficiency. Our future work includes a better way to generate the blob image for tracking. Since the scale estimation algorithm is relatively independent from the target locating procedure of the tracker, the employment of the algorithm for other visual trackers is also worth studying.

7 Acknowledgment

This work is supported by the National Natural Science Foundation of China (grant nos. 61175032, 61302154 and 61304096).

8 References

1 Benfold, B., Reid, I.: 'Stable multi-target tracking in real-time surveillance video'. IEEE Conf. Computer Vision and Pattern Recognition, Providence, USA, June 2011, pp. 3457–3464
2 Cao, X., Gao, C., Lan, J., Yuan, Y., Yan, P.: 'Ego motion guided particle filter for vehicle tracking in airborne videos', *Neurocomputing*, 2014, **124**, pp. 168–177
3 Oikonomidis, I., Kyriazis, N., Argyros, A.A.: 'Tracking the articulated motion of two strongly interacting hands'. IEEE Conf. on Computer Vision and Pattern Recognition, Providence, USA, June 2012, pp. 1862–1869
4 Prisacariu, V.A., Reid, I.: '3D hand tracking for human computer interaction', *Image Vis. Comput.*, 2012, **30**, (3), pp. 236–250
5 Grigorescu, S.M., Pozna, C.: 'Towards a stable robotic object manipulation through 2D-3D features tracking', *Int. J. Adv. Robot. Syst.*, 2013, **10**, p. 200
6 Avidan, S.: 'Support vector tracking', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2004, **26**, (8), pp. 1064–1072
7 Yilmaz, A., Javed, O., Shah, M.: 'Object tracking: a survey', *ACM Comput. Surv. (CSUR)*, 2006, **38**, (4), pp. 1–45
8 Wu, Y., Lim, J., Yang, M.: 'Online object tracking: a benchmark'. IEEE Conf. Computer Vision and Pattern Recognition, Oregon, USA, June 2013, pp. 2411–2418
9 Comaniciu, D., Ramesh, V., Meer, P.: 'Kernel-based object tracking', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2003, **25**, (5), pp. 564–577
10 Collins, R.T.: 'Mean-shift blob tracking through scale space'. IEEE Conf. Computer Vision and Pattern Recognition, 2003, pp. II: 234–240
11 Zivkovic, Z., Krose, B.: 'An EM-like algorithm for color-histogram-based object tracking'. IEEE Conf. Computer Vision and Pattern Recognition, 2004, p. I–798
12 Ning, J., Zhang, L., Zhang, D., Wu, C.: 'Scale and orientation adaptive mean shift tracking', *IET Comput. Vis.*, 2012, **6**, (1), pp. 52–61
13 Jiang, Z., Li, S., Gao, D.: 'An adaptive mean shift tracking method using multiscale image'. Proc. Int. Conf. Wavelet Analysis and Pattern Recognition, Beijing, China, November 2007, pp. 1060–1066
14 Vojir, T., Noskova, J., Matas, J.: 'Robust scale-adaptive mean-shift for tracking', *Image Anal.*, 2013, **7944**, pp. 652–663
15 Bradski, G.R.: 'Real time face and object tracking as a component of a perceptual user interface'. IEEE Proc. Applications of Computer Vision, Princeton, USA, October 1998, pp. 214–219
16 Liang, D., Huang, Q., Jiang, S., Yao, H., Gao, W.: 'Mean-shift blob tracking with adaptive feature selection and scale adaptation'. IEEE Int. Conf. Image Processing, San Antonio, USA, September 2007, pp. 369–372
17 Collins, R.T., Liu, Y., Leordeanu, M.: 'Online selection of discriminative tracking features', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2005, **27**, (10), pp. 1631–1643
18 Viola, P., Jones, M.: 'Rapid object detection using a boosted cascade of simple features'. IEEE Conf. Computer Vision and Pattern Recognition, 2001, p. I–511
19 Jeyakar, J., Babu, R.V., Ramakrishnan, K.R.: 'Robust object tracking with background-weighted local kernels', *Comput. Vis. Image Underst.*, 2008, **112**, (3), pp. 296–309
20 Visual Track Benchmark. Available at http://www.cvlab.hanyang.ac.kr/tracker_benchmark_v10.html, accessed November 2014
21 Tracking Dataset. Available at http://www.cmp.felk.cvut.cz/~vojirtom/dataset, accessed November 2014
22 Suryanto, Kim, D., Kim, H., Ko, S.: 'Spatial color histogram based center voting method for subsequent object tracking and segmentation', *Image Vis. Comput.*, 2011, **29**, (12), pp. 850–860