Mean-Shift Object Tracking with Discrete and Real AdaBoost Techniques

Hendro Baskoro, Jun-Seong Kim, and Chang-Su Kim

An online mean-shift object tracking algorithm, which consists of a learning stage and an estimation stage, is proposed in this work. The learning stage selects the features for tracking, and the estimation stage composes a likelihood image and applies the mean shift algorithm to it to track an object. The tracking performance depends on the quality of the likelihood image. We propose two schemes to generate and integrate likelihood images: one based on the discrete AdaBoost (DAB) and the other based on the real AdaBoost (RAB). The DAB scheme uses tuned feature values, whereas RAB estimates class probabilities, to select the features and generate the likelihood images. Experiment results show that the proposed algorithm provides more accurate and reliable tracking results than the conventional mean shift tracking algorithms.

Keywords: Mean-shift, blob tracking, object tracking, adaptive boosting (AdaBoost), likelihood image.

Manuscript received July 1, 2008; revised Feb. 9, 2009; accepted Apr. 9, 2009.

This work was supported partly by the Ministry of Knowledge Economy, Rep. of Korea, under the Information Technology Research Center support program supervised by the Institute of Information Technology Advancement (grant number IITA-2009-C1090-0902-0017) and partly by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. R01-2008-000-20292-0).

Hendro Baskoro (phone: +82 2 3290 3806, email: hendro.baskoro@hotmail.com) is with Digital Media Division, LG Electronics, Seoul, Rep. of Korea.

Jun-Seong Kim (email: junssi153@korea.ac.kr) and Chang-Su Kim (phone: +82 2 3290 3217, email: cskim@ieee.org) are with the School of Electrical Engineering, Korea University, Seoul, Rep. of Korea.

I. Introduction

The objective of object tracking is to estimate the trajectory of an object in an image plane as it moves around a scene [1]. In other words, a tracker estimates the location of a target object throughout the frames in a video sequence.

Object tracking is an essential element in various computer vision applications, including motion-based object recognition, human-computer interaction, and automatic vehicle navigation [1]. In [2], tracking is employed to enhance virtual reality experience by using human body movements to navigate and interact with a virtual world. Tracking is also important in surveillance systems [3]. Since surveillance systems produce a large amount of video data, an automatic algorithm is needed to index and retrieve this data.

Mean-shift blob tracking is an approach to object tracking which estimates an object's location in a frame by performing the mean-shift algorithm on a likelihood image of the frame [4]-[7]. A likelihood image is a gray scale image, where the brightness of a pixel represents the likelihood of its being a part of the object. A good likelihood image should have a bright area around the object being tracked, while the remaining area should be dark.

The ensemble tracking algorithm in [7] attempts to obtain a good likelihood image for mean-shift blob tracking. It uses multiple weak classifiers to divide object pixels from background pixels in a feature space. The weak classifiers are combined into a strong classifier, using the adaptive boosting (AdaBoost) technique [8]. The strong classifier is then used to construct a likelihood image.

An alternative approach is to use multiple features, instead of a single feature space. Collins and Liu [5] proposed using linear combinations of RGB color channels as a feature pool

and selecting the features for tracking based on two-class variance ratios. A likelihood image is generated for each selected feature, and the mean-shift algorithm is applied to estimate the object location. Each likelihood image produces an estimated location, and the median of these locations is taken as the final object location. In the mean-shift algorithm, the reliability of the estimated location greatly depends on the quality of the likelihood image. However, in [5], there are only a few likelihood images to track an object, and the tracker sometimes fails simply because there is no good likelihood image. In [6], features are selected using Bayesian error rates, and likelihood images are weighted based on the error rates to obtain a higher quality likelihood image.

In this work, we propose selecting features from a feature pool using the AdaBoost technique and combining likelihood images for different features using confidence values. Specifically, we propose two online tracking schemes: one based on the discrete AdaBoost (DAB) [9] and the other based on the real AdaBoost (RAB) [8]. These schemes differ in the feature selection, the modeling of object and background, and the generation and integration of likelihood images. Note that both the proposed algorithm and ensemble tracking [7] use the AdaBoost technique to improve the performance of mean-shift blob tracking. However, the main difference is that the proposed algorithm uses multiple feature spaces with the same classifier, whereas ensemble tracking combines multiple classifiers in a single feature space. Experimental results show that the proposed algorithm provides higher quality likelihood images and achieves better tracking accuracy than the conventional algorithms in [5]-[7].

The rest of this paper is organized as follows. Section II gives an overview of the proposed algorithm, which consists of a learning stage and an estimation stage. The learning stage is described in section III, and the estimation stage is described in section IV. Section V presents and discusses experimental results. Finally, section VI concludes the paper.

II. Overview

The proposed tracking algorithm consists of two main stages: a learning stage and an estimation stage. In the learning stage, we select features and generate models for an object and the background. In the estimation stage, we compose a likelihood image, using the features and the models, and then apply the mean-shift algorithm to the likelihood image to estimate the object's location. These two stages are interleaved to adapt to changes in the object's appearance. The proposed algorithm belongs to the online tracking category, since the selected features and the models are continuously updated from the input video sequence itself.

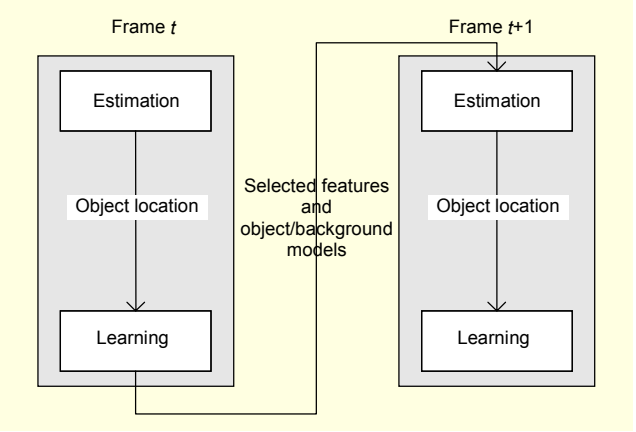


Fig. 1. The estimation stage and the learning stage are executed iteratively. For the first frame, the estimation stage is not performed, since the object location is assumed to be known. For the other frames, the object locations from the estimation stage are used to select features and generate the models of object and background in the learning stage.

In this work, we execute the estimation stage and the learning stage for every frame by feeding the object location, produced in the estimation stage, to the learning stage as shown in Fig. 1. For the first frame, only the learning stage is carried out, since we assume that the object location in the first frame is available from a user or from an object detection scheme. We also assume that the object being tracked is always visible. If there is an occlusion, then the algorithm should be re-initialized when the object becomes visible again. Starting from the second frame, the estimation stage is performed and followed by the learning stage. However, note that the learning stage does not need to be applied for every frame, since the changes in object and background appearance are gradual in general. In other words, the learning stage can be applied for a selected set of frames by only using simple frame skipping [5] or the Gaussian model [6].

As in [5], the combinations of RGB channels are used as a feature pool F:

$$F = \{w_1 R + w_2 G + w_3 B \mid w_1, w_2, w_3 \in \{-2, -1, 0, 1, 2\}\}. \quad (1)$$

There can be 5^3 combinations, but we remove the abnormal combinations, where $(w_1, w_2, w_3) = k(0, 0, 0)$, and the redundant combinations, where $(w_1, w_2, w_3) = k(w'_1, w'_2, w'_3)$. Thus, the total number of features in the pool is reduced to 49.

III. Learning Stage

In the learning stage, we assume that the object location is known from a user or from the estimation stage. We select the features for tracking from the feature pool, and make the

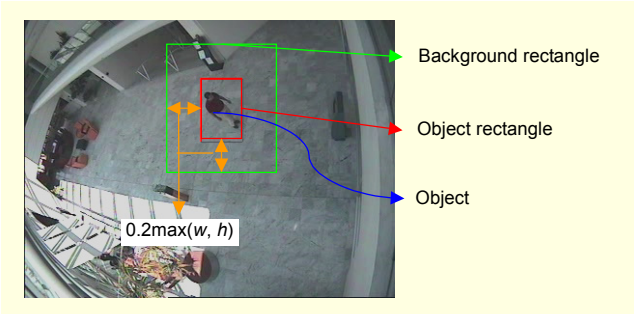


Fig. 2. The center-surround approach is used to take samples of object and background pixels. Here, w and h are the width and the height of the object rectangle.

models of the object and background using those features. The features and the models are passed to the estimation stage for the next frame.

We represent the object as a rectangle of fixed size and orientation. We adopt the center-surround approach [5] in sampling the object and background pixels. As shown in Fig. 2, it expands the object rectangle in order to form the background rectangle, from which the samples of background pixels are taken. In our implementation, the margin between the sides of the object rectangle and the background rectangle is set to 0.2max(w, h), where w and h denote the width and the height of the object rectangle.

The pixels within the object rectangle and the background rectangle are employed as training examples in the AdaBoost framework [8]-[10], and features are selected to minimize the probability of incorrectly classifying the training pixels into the object or background class.

The features trained from the information in the current frame can be used to track the object in the next frame because the object movements between consecutive frames are usually small. Unlike the approach using a fixed set of features, the features in an online tracker only need to be locally and temporarily discriminative [5].

We use DAB [9] or RAB [8] to build an effective likelihood image from simple feature images. The feature selection and the model generation are carried out concurrently. Given a feature, we model the object and the background and evaluate the classification performance on training pixels. Features are iteratively selected by evaluating all features, choosing the best features, and updating the weights of training pixels. More specifically, we assign a weight to each pixel and choose the feature that minimizes the weighted misclassification error. Before the next iteration, we decrease the weights of correctly classified pixels and increase the weights of misclassified pixels. Thus, independence among the selected features is promoted and the performance of the selected features as a whole is improved.

1. DAB Learning

The weights of training pixels are initialized according to their spatial distances from the center of the object rectangle. Specifically, the weight for a pixel in the object rectangle is initialized by

$$1-2\times\frac{\text{distance from center}}{\text{diagonal of object rectangle}}, \tag{2}$$

and the weight for a pixel in the background area is set by

$$1-2\times\frac{\text{distance from center}}{\text{diagonal of background rectangle}}. \tag{3}$$

These weights are then normalized so that the sum of weights in the object area is 0.5 and the sum of weights in the background area is 0.5.

Each feature is evaluated based on how well it classifies the training pixels. The classification is performed as follows. First, we compute the normalized histograms for object pixels, $p(k)$, and background pixels, $q(k)$, in the feature image. In DAB, for simplicity, we do not consider the weights of pixels in the computation of the histograms. Second, we compute the tuned feature value $L(k)$, which is defined as the log ratio of the object histogram and the background histogram:

$$L(k)=\log\frac{\max\{p(k),\delta\}}{\max\{q(k),\delta\}}, \tag{4}$$

where δ is a small fixed number to avoid computational overflows. Third, we use zero as the classification threshold. Pixels with positive tuned feature values are classified as the object, while pixels with negative values are classified as the background. The optimal threshold may not be zero. Although it can be found by examining the misclassification error for every possible threshold [6], we use the threshold 0 for simplicity.

In the DAB learning stage, we take the positions of the object and background rectangles as input and produce a selection of features with confidence values. The algorithm to select T features from the feature pool F is carried out as follows:

Step 1. Initialize the weights of object pixels, $w^1_{\text{obj},i}$, and the weights of background pixels, $w^1_{\text{bg},j}$, and normalize them so that the sums of object and background weights are 0.5 and 0.5, respectively.

Step 2. Repeat steps 2.1 to 2.4 for $t=1,\cdots,T$.

Step 2.1. For each feature f in the pool F, compute the tuned feature values for the training pixels, classify the pixels, and then evaluate the misclassification error with respect to $w^t_{\text{obj},i}$ and $w^t_{\text{bg},j}$

$$e_f^t = \sum_{i \in I} w_{\text{obj},i}^t + \sum_{j \in J} w_{\text{bg},j}^t,$$

where I and J are, respectively, the sets of object pixels and background pixels that are misclassified.

Step 2.2. Select the best feature f_t that has the lowest error

$$e_t = \min_f e_f^t.$$

The confidence c_t of the selected feature is defined as

$$c_t = \log \frac{1-e_t}{e_t}.$$

Step 2.3. Reduce the weights of correctly classified pixels by

$$w_{\text{obj},i}^{t+1} = w_{\text{obj},i}^t \frac{e_t}{1-e_t} \quad \text{and} \quad w_{\text{bg},j}^{t+1} = w_{\text{bg},j}^t \frac{e_t}{1-e_t}.$$

Then, normalize the weights

$$w_{\text{obj},i}^{t+1} = \frac{w_{\text{obj},i}^{t+1}}{2\sum_i w_{\text{obj},i}^{t+1}} \quad \text{and} \quad w_{\text{bg},j}^{t+1} = \frac{w_{\text{bg},j}^{t+1}}{2\sum_j w_{\text{bg},j}^{t+1}}.$$

Step 2.4. Remove the selected feature f_t from the pool F

$$F = F - \{f_t\}.$$

In DAB, the object in a selected feature f_t is differentiated from the background through the tuned feature values $L_t(k)$. These tuned feature values are stored to generate likelihood images in the estimation stage for the next frame. To summarize, the results of the DAB learning stage are the selected features f_t with the corresponding confidences c_t and the tuned feature values $L_t(k)$.

2. RAB Learning

In RAB, the weights of training pixels are initialized in the same way as in DAB as described in section III.1.

Again, a feature is evaluated based on how well it classifies training pixels. Whereas DAB makes a hard decision whether a pixel belongs to the object or background class by comparing its tuned feature value with a threshold, RAB softly expresses the classification result with a real number. We first compute the normalized object histogram $p(k)$ and the normalized background histogram $q(k)$ using the weights of training pixels. Specifically, instead of counting the number of pixels quantized to bucket k, we add up the weights of those pixels and normalize the sum to compute $p(k)$ or $q(k)$. Then, the classification result is expressed as a real value ranging from 0 to 1, given by

$$r(k) = \frac{p(k)}{p(k)+q(k)}, \tag{5}$$

which is the estimated probability that a pixel belongs to the object class. Note that the pixels whose feature values belong to the same histogram bucket are tagged with the same real value. The confidence of the bucket-by-bucket classification is defined as

$$c(k) = \frac{1}{2} \log \frac{r(k)}{1-r(k)}. \tag{6}$$

The confidence $c(k)$ approaches ∞ or $-\infty$ as the probability $r(k)$ becomes close to 1 or 0, respectively. On the other hand, $c(k)$ is 0, when $r(k)$ is 0.5. That is, a pixel has equal probability of belonging to the object class and the background class.

The RAB learning algorithm to select T features from the feature pool F is carried out as follows:

Step 1. Initialize the weights of training pixels, w_i^1, and normalize them as described in section III.1.

Step 2. Repeat steps 2.1 to 2.4 for $t = 1, \cdots, T$.

Step 2.1. For each available feature f from the pool F, compute the normalized histograms $p_f^t(k)$ and $q_f^t(k)$ using the weights w_i^t. A pixel whose feature value is quantized to bucket k belongs to the object class with the estimated probability

$$r_f^t(k) = \frac{p_f^t(k)}{p_f^t(k)+q_f^t(k)}.$$

The confidence value of bucket k is given by

$$c_f^t(k) = \frac{1}{2} \log \frac{r_f^t(k)}{1-r_f^t(k)}.$$

Step 2.2. Select the best feature f_t that has the maximum sum of absolute confidences

$$f_t = \arg\max_f \sum_k \left| c_f^t(k) \right|.$$

Let $c_t(k)$ denote the confidence value of the selected feature f_t for notational simplicity.

Step 2.3. Update the weights

$$w_i^{t+1} = w_i^t \exp\left[-y_i c_t(k)\right], \tag{7}$$

where the feature value of pixel i is assumed to be quantized into bucket k. Here, y_i=1 if pixel i belongs to the object and y_i=-1 otherwise. Note that an object pixel is correctly classified if it has a positive confidence value $c_t(k)$. Also, a correctly classified background pixel has a negative confidence. Thus, the purpose of y_i in (7) is to reduce the weights of correctly classified pixels. Then, normalize the weights by

$$w_i^{t+1} = \frac{w_i^{t+1}}{\sum_i w_i^{t+1}}.$$

Step 2.4. Remove the selected feature f_t from the pool F

$$F = F - \{f_t\}.$$

Whereas DAB computes a confidence value for a whole feature, RAB specifies a confidence value for each histogram bucket of a feature. In RAB, the selected features f_t and the corresponding confidence values $c_t(k)$ are passed to the estimation stage of the next frame.

IV. Estimation Stage

The object location is estimated by generating likelihood images based on the selected features in the last learning stage, combining them into a final likelihood image, and then performing the mean shift algorithm to the final likelihood image. The schemes to generate and combine likelihood images are different for DAB and RAB.

1. DAB Estimation

In the DAB learning stage, features f_t, $1 \le t \le T$, are selected, and their confidence values c_t and tuned feature values $L_t(k)$ are computed. Using this data, the estimation stage tracks the object location.

First, we generate a binary likelihood image for each selected feature f_t as shown in Fig. 3(a). The feature value of a pixel is quantized into bucket k. If the tuned feature value $L_t(k)$ is less than the threshold 0, the pixel tends to be of the background class and is assigned the value of 0 in the likelihood image. On the other hand, if $L_t(k)$ is positive, the pixel tends to be of the object class and is assigned the value of 1. This step is repeated for all selected features.

Second, we combine the T likelihood images as shown in Fig. 3(b). The likelihood image for feature f_t is weighted by the confidence value c_t, and all the weighted likelihood images are summed up to make a single image. Then, pixel values less than $0.5 \Sigma c_t$ are set to zero and the remaining values are scaled to the range [0, 255]. The obtained image serves as the final likelihood image.

Third, we apply the mean-shift algorithm [4], [11], [12] to the final likelihood image to track the center of the object rectangle. The estimated center is initialized by the center of the object rectangle in the previous frame and is iteratively updated according to the mean-shift vectors, which point to the local maxima of the likelihood image.

The mean-shift vector $\vec{m}$ is given by

$$\vec{m} = \frac{\sum_{i \in R} \vec{x}(i) l(i)}{\sum_{i \in R} l(i)}, \quad (8)$$

where R denotes a local window around the current center, $\vec{x}(i)$ is the vector from the current center to pixel i, and $l(i)$ is the likelihood value of pixel i. The size of R is set to about 90% of the object rectangle. The mean-shift algorithm terminates when the mean-shift vector becomes zero or the maximum number of iterations is reached. In this work, the maximum number of iterations is set to 20.

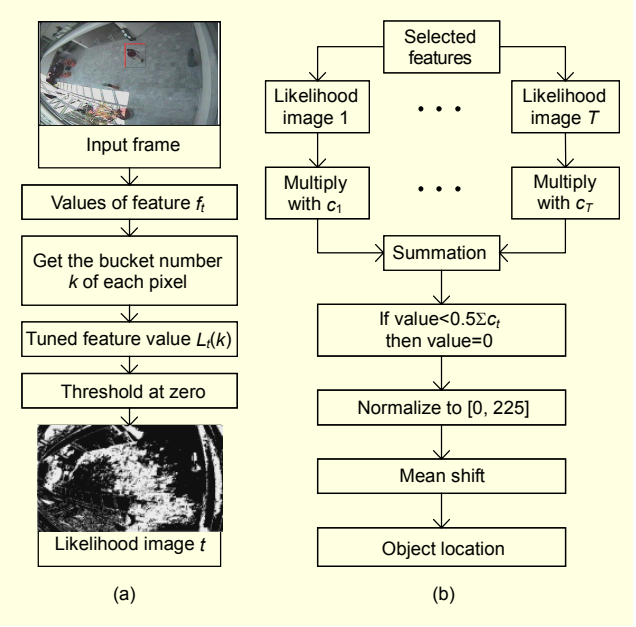


Fig. 3. (a) Generation of a likelihood image and (b) integration of likelihood images in the DAB estimation stage.

2. RAB Estimation

As shown in Fig. 4(a), we generate a likelihood image for each feature f_t. First, we quantize the feature value of a pixel to bucket k. The confidence value $c_t(k)$ of the bucket becomes the brightness of the pixel in the likelihood image. This is repeated for all selected features.

Then, as shown in Fig. 4(b), we sum the T likelihood images and normalize the brightness into the range [0, 255] to make the final likelihood image. During the normalization, negative values are set to zero.

Finally, as in section IV.1, we apply the mean-shift algorithm to the final likelihood image to track the object location.

V. Experiments

We compared the performance of the proposed algorithm with that of Collins and Liu's algorithm [5], Liang's algorithm [6], and the ensemble tracking algorithm [7]. The learning stage was performed in every frame without skipping. For the learning stage, the object location in the first frame was provided by the user, while the locations in the other frames were provided by the estimation stage.

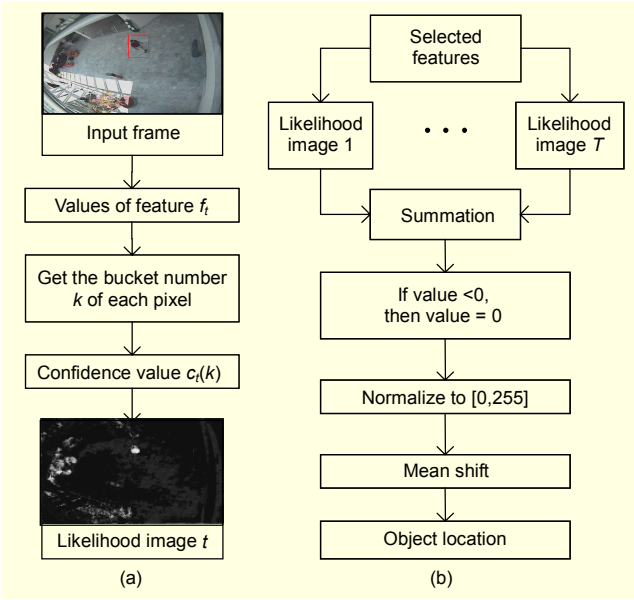


Fig. 4. (a) Generation of a likelihood image and (b) the integration of likelihood images in the RAB estimation stage.

Five test sequences were used for the experiments: "ice," "browse," "football," "coastguard," and "clip." The resolutions of these sequences vary from 320 × 240 to 384 × 288 pixels. Every test sequence consists of 50 frames. We intentionally chose those frames to track objects that are always visible and have an approximately fixed size throughout the sequence. Each sequence was tested with two different numbers of features. Thus, there was a total of 10 tests. The same mean shift parameters were used for all the tests.

Figures 5 to 9 show examples of the tracking results. The test sequences differ in object rigidity, the number of background pixels included in the object rectangle, the speed of object motion, and the camera movement. For the "ice," "browse," and "clip" sequences, the camera movements are unnoticeable and the objects move slowly. Object tracking in the "clip" sequence was generally easy because the object has a very different color from the background. The tests on the "ice" and "browse" sequences were relatively difficult, since the objects are not rigid and the objects and the backgrounds have similar colors. Furthermore, the object rectangle in the "browse" sequence contains a lot of background pixels because the object orientation changes from diagonal to vertical. Camera movements exist in the "football" and "coastguard" sequences. Tracking in the "football" sequence was the most difficult test because the tracked object in the "football" sequence exhibits very fast motions with motion blurs. Moreover, more than one third of the tracked object has similar colors to the background. In the "coastguard" sequence, the upper part of the object

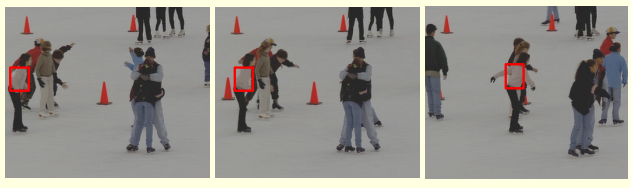


Fig. 5. Tracking results of the DAB algorithm from the "ice" sequence at frames 2, 11, and 50.

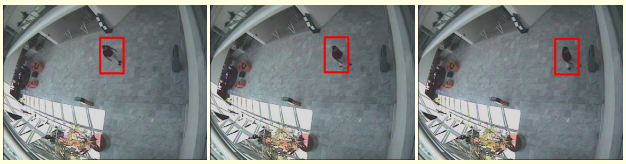


Fig. 6. Tracking results of the RAB algorithm from the "browse" sequence at frames 1, 25, and 50.

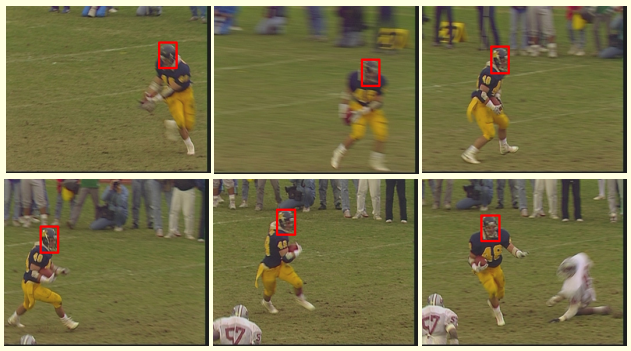


Fig. 7. Tracking results of the DAB algorithm from the "football" sequence at frames 2, 10, 18, 26, 34, and 42.

Fig. 8. Tracking results of the RAB algorithm from the "coastguard" sequence at frames 2, 25, and 50.

Fig. 9. Tracking results of the DAB algorithm from the "clip" sequence at frames 2, 25, and 50.

rectangle contains many background pixels.

The accuracy of an estimated object location is measured by the Euclidean distance between the estimation result and the

Table 1. Euclidean distance performance. L is Liang's method [6], CL is Collins and Liu's algorithm [5], ET is the ensemble tracking algorithm [7], and DAB and RAB are the proposed algorithms. Yellow boxes represent tracking failures, and bold numbers highlight the best results.

Sequence	Number of features	Euclidean distance (pixels)																			
		Maximum					Minimum					Mean					Variance				
		L	CL	ET	DAB	RAB	L	CL	ET	DAB	RAB	L	CL	ET	DAB	RAB	L	CL	ET	DAB	RAB
Ice (352×288)	1	**5.8**	6.71	147.3	**5.8**	8.94	**0**	**0**	8	**0**	**0**	3	2.749	77.63	2.977	**2.33**	**2.38**	3.034	1705	2.492	4.231
	3	5.83	6.71	146.5	**5**	8.06	**0**	**0**	8.062	**0**	**0**	3.12	2.836	73.52	2.503	**1.72**	2.791	3.539	1553	**1.81**	2.718
Browse (384×288)	5	84.5	24	14.56	14.6	**13**	**3**	4.243	5	3.6	4.243	45.03	12.3	9.537	**9.29**	9.343	543	28.02	5.727	6.568	**4.7**
	9	84.5	24.4	15.52	13.9	**13**	**3**	4.243	5	3.6	5	45.03	12.4	9.449	**9.06**	9.727	543	**31.79**	6.758	6.201	**4.59**
Football (352×288)	3	195	24	44.38	**9.1**	12.2	5.385	**0**	1	1	**0**	127.1	4.827	24.79	**3.64**	4.099	3284	27.55	155.3	**4.41**	7.953
	5	196	125	44.38	39.1	**13**	5.385	**0**	1	**0**	1	128.1	40.85	25.31	17.42	**4.24**	3305	1804	157.8	187.4	**8.32**
Coastguard (352×288)	1	15.3	**8.1**	10.2	9.06	**8.1**	0	1.414	3.606	3	3.606	7.326	**5.64**	5.885	6.011	6.067	5.05	3.496	1.541	4.32	**1.25**
	3	18.4	8.1	12.65	**7.3**	**7.3**	2	**1.4**	3.162	3	4	7.702	5.718	**5.4**	5.693	6.083	5.966	3.556	2.12	1.39	**0.9**
Clip (320×240)	1	5	**2.2**	**2.2**	3.16	**2.2**	**0**	**0**	**0**	**0**	**0**	2.636	**1.08**	1.297	1.783	1.194	1.802	**0.28**	0.386	0.527	0.357
	3	5	**2.2**	**2.2**	3.16	**2.2**	**0**	**0**	**0**	**0**	**0**	2.623	**1**	1.05	1.697	1.165	1.976	**0.33**	0.396	0.393	0.343

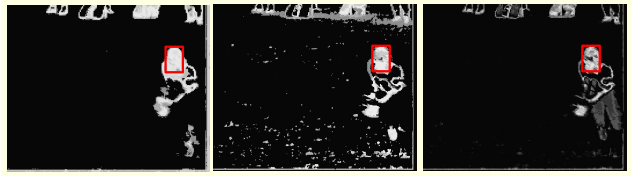
ground truth information provided by a human observer. Table 1 compares the Euclidean distance performance of the tested algorithms. For each test, we measured the maximum, minimum, and mean distances, as well as the variance of distances over the sequence. In Table 1, bold numbers highlight the best results, and yellow boxes indicate tracking failures. Liang's algorithm [6] yielded failures on the "browse" and "football" sequences; however, note that we did not use the scale adaptation scheme in [6] for a fair comparison with the other algorithms. Liang's algorithm provides longer Euclidean distances than the other algorithms in general. Collins and Liu's algorithm failed on the "football" sequence only, and yielded shorter mean distances than Liang's algorithm. However, we see that the proposed DAB and RAB algorithms are generally better than the Collins and Liu's algorithm in terms of the mean distance and the distance variance. In particular, the proposed DAB and RAB algorithms yield significantly lower distance variances than Collins and Liu's algorithm. A lower variance indicates that the estimated trajectory matches the true trajectory more reliably. Based solely on the number of failures, we conclude that RAB is the most reliable algorithm, and that the reliability of DAB is comparable with that of Collins and Liu's algorithm.

Table 1 includes the performance of the ensemble tracking algorithm in [7] also for comparison. While Liang's algorithm, Collins and Liu's algorithm, and the proposed DAB and RAB schemes select a number of features adaptively during tracking, the ensemble tracking algorithm uses a fixed feature space and constructs the strong classifier adaptively by combining multiple weak classifiers. The ensemble tracking fails on the "ice" sequence. This is because it uses only the RGB values and the local histogram of oriented gradients as a feature vector. Since the object has RGB values and textures that are similar to those of the background, ensemble tracking cannot construct a good classifier on the feature space and therefore fails. Similarly, on the "football" sequence, the tracking window deviates from the helmet to the shoulder of the player, which has similar colors and textures. On the other hand, the proposed algorithm combines the RGB values to form a feature pool and selects features adaptively so that it can discriminate the object from the background successfully. Except for the "ice" and "football" sequences, the ensemble tracking algorithm and the proposed algorithm achieve comparable distance performance.

If more features are used for the same sequence, the proposed algorithm achieves better tracking results in general, whereas the performance of Liang's algorithm and Collins and Liu's algorithm gets worse in many cases. Let T denote the number of features. In Liang's algorithm and Collins and Liu's algorithm, features are ranked according to their individual performance, and the best T features are selected without considering the inter-feature relationships. For example, let us consider the cases of T=1 and T=2. If the second best feature is not good enough, its addition may degrade the overall performance, and using one feature may be better than using two features. On the other hand, in the proposed algorithm, features are selected by the AdaBoost techniques iteratively. At each iteration, pixels which are already well classified with the existing features are assigned lower weights, and misclassified pixels are assigned higher weights in the selection of the next feature. In this way, the proposed algorithm selects the T

(a) (b) (c)

Fig. 10. Likelihood images for frame 3 of the "ice" sequence, obtained by (a) Collins and Liu's algorithm [5], (b) DAB, and (c) RAB.



(a) (b) (c)

Fig. 11. Likelihood images for frame 3 of the "football" sequence, obtained by (a) Liang's algorithm [6], (b) DAB, and (c) RAB.

(a) (b) (c)

Fig. 12. Likelihood images for frame 24 of the "football" sequence, obtained by (a) Collins and Liu's algorithm [5], (b) DAB, and (c) RAB.

features that compensate the weaknesses of one another. By using more features, the RAB scheme reduces the mean and maximum distances in two sequences and reduces the variances in four sequences. Also, by using more features, the DAB scheme reduces the mean distances and the variances in four sequences while reducing the maximum distances in three sequences.

Figures 10 to 12 show several likelihood images for visual comparison. In these images, red rectangles represent the ground truth object rectangles obtained by human observation. Figure 10(a) is the single best likelihood image, employed in Collins and Liu's algorithm. We see that there are dark pixels inside the object rectangle especially in the upper left part, whereas the lower right region outside the rectangle is bright. In this condition, the mean-shift algorithm deviates from the correct position to the lower right. A similar condition is observed in the likelihood image of the DAB algorithm, as shown in Fig. 10(b), but the pixels inside the object rectangle are much brighter than those in Fig. 10(a). Thus, the deviation effect is alleviated. Figure 10(c) is a likelihood image of the proposed RAB algorithm, where the brightness is generally lower. However, the brightness difference between the inside and the outside of the object rectangle is bigger; thus, a more precise tracking result can be achieved. Similar tendencies can be observed in the likelihood images for the "football" sequence in Figs. 11 and 12.

Table 2. Processing times: L is Liang's method [6], CL is Collins and Liu's algorithm [5], and DAB and RAB are the proposed schemes. Yellow boxes represent tracking failures, and bold numbers highlight the best results.

Sequence	Number of features	Processing time (s)			
		L	CL	DAB	RAB
Ice (352×288)	1	**4.03**	4.24	4.28	4.16
	3	**4.75**	4.83	10.16	9.31
Browse (384×288)	5	9.14	**6.16**	21.16	19.08
	9	11.75	**8.13**	35.31	31.88
Football (352×288)	3	6.47	**4.94**	11.42	10.22
	5	7.13	5.56	16.28	**14.86**
Coastguard (352×288)	1	11.13	**5.08**	5.88	5.49
	3	11.33	**5.84**	14.8	13.02
Clip (320×240)	1	4.75	4.94	**3.59**	5.8
	3	**5.25**	5.42	8.56	15.02

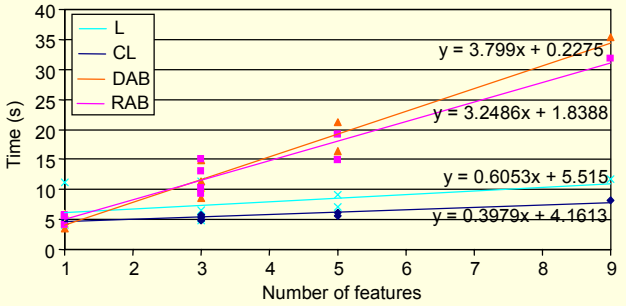


Fig. 13. Processing times of Liang's algorithm (L) [6], Collins and Liu's algorithm (CL) [5], and the proposed DAB and RAB schemes in terms of the number of features.

Table 2 summarizes the processing times to obtain the results in Table 1. Also, Fig. 13 plots the processing times and their linear regression lines in terms of the number of features. The processing times of the ensemble tracking algorithm are not included in Table 2 and Fig. 13 because we have not optimized the code of the algorithm, which uses a lot of matrix multiplications in the learning of classifiers. Thus, the processing times of the ensemble tracking algorithm are longer than those of the proposed algorithms. However, after optimization, we believe that ensemble tracking will demand complexities comparable to those of the proposed schemes. We

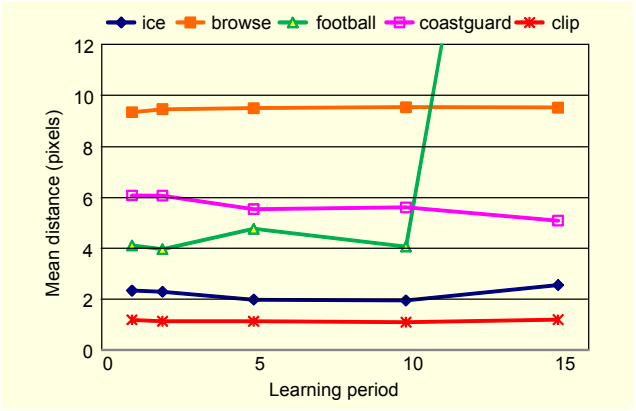


Fig. 14. Distance performance of the proposed RAB algorithm according to the learning period. The numbers of features are set to 3, 9, 5, 3, and 3 for the "ice," "browse," "football," "coastguard," and "clip" sequences, respectively.

see that the complexities of the proposed DAB and RAB schemes increase more quickly than those of conventional algorithms when more features are used for tracking. This is largely because the proposed schemes choose T features using the AdaBoost learning methods, which are iterative and demand relatively high complexities. On the other hand, the conventional algorithms are simpler, since they rank features based on a feature selection criterion and choose the T best features simultaneously. However, note that by the careful selection and integration of features, the proposed schemes provide more reliable tracking results than the conventional algorithms.

The learning stage does not need to be carried out at every frame. For example, to reduce the processing complexity, we can perform the learning stage at every k-th frame only, where k denotes the learning period. Figure 14 shows how the distance performance varies according to the learning period. We see that the distance performance is not sensitive to the learning period, since the objects and the backgrounds change their appearance gradually in these sequences, and the learned information for a certain frame can be used to track objects in the subsequent frames. Moreover, on the "coastguard" sequence, the performance is slightly degraded when learning is too frequently performed. This is due to drift errors: as we learn features and models online, inaccuracies in a frame propagate to the subsequent frames. However, in general, the proposed algorithm achieves better performance, as we perform the learning stage more frequently. The only exception is the "football" sequence, which contains fast motions and experiences significant deformations of the object. When the learning period is larger than 10, the proposed algorithm fails to track the object in the "football" sequence correctly. The learning period should be selected carefully by considering the tradeoff relationship between the processing complexity and the tracking accuracy.

The proposed algorithm has several limitations. Since we use only color information, the performance degrades when the object and the background have similar color distributions. The addition of textures and spatial information into the feature vector will improve tracking performance. Also, since an object is represented coarsely by an enclosing rectangle, background pixels are mislabeled as belonging to the object, contaminating the object model. However, we have found that this is not a problem, provided that there are similar pixels within the background window. Then, in the AdaBoost learning stage, these pixels are assigned lower weights, and the other pixels are treated as more important in the feature selection. But, when the background has irregular colors and textures, the rectangular window is not sufficient, and a tighter window is required. Another limitation is that the proposed algorithm does not consider the occlusion of objects. An occlusion handling procedure should be incorporated for the proposed algorithm to be used reliably in practical applications.

VI. Conclusion

In this work, we proposed two mean-shift object tracking algorithms: one based on DAB and the other based on RAB. DAB uses tuned feature values, whereas RAB estimates class probabilities, to select features and generate likelihood images. By employing the AdaBoost framework, the proposed algorithms can enhance the independence among selected features and compose high quality likelihood images. Experiment results demonstrated that the proposed algorithm can provide more accurate and reliable tracking results than Liang's algorithm [6], and Collins and Liu's algorithm [5]. Future research issues include the development and implementation of a fast learning algorithm to reduce the processing time for online tracking.

References

[1] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys,* vol. 38, no. 4, Dec. 2006.

[2] C.R. Wren et al., "Pfinder: Real-Time Tracking of the Human Body," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, July 1997, pp. 780-785.

[3] W. Hu et al., "Semantic-Based Surveillance Video Retrieval," *IEEE Trans. on Image Processing*, vol. 16, no. 4, Apr. 2007, pp. 1168-1181.

[4] R.T. Collins, "Mean-Shift Blob Tracking Through Scale Space," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, June 2003, pp. 234-240.

[5] R.T. Collins and Y. Liu, "On-Line Selection of Discriminative Tracking Features," *Proc. International Conference on Computer Vision*, vol. 1, Oct. 2003, pp. 346-352.

[6] D. Liang et al., "Mean-Shift Blob Tracking with Adaptive Feature Selection and Scale Adaptation," *Proc. IEEE International Conference on Image Processing*, vol. 3. Sept.-Oct. 2007, pp. 369-372.

[7] S. Avidan, "Ensemble Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, Feb. 2007, pp. 261-271.

[8] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics*, vol. 28, no. 2, Apr. 2000, pp. 337-407.

[9] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Dec. 2001, pp. 511-518.

[10] Yi Hu et al., "Real-Time Apartment Building Detection and Tracking with AdaBoost Procedure and Motion-Adjusted Tracker," *ETRI Journal*, vol. 30, no. 2, Apr. 2008, pp 338-340.

[11] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, May 2002, pp. 603-619.

[12] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-Based Object Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, May 2003, pp. 564-577.

Hendro Baskoro received the BS degree in computer engineering from Bandung Institute of Technology (ITB), Indonesia, in July 2005, and the MS degree in electronics and computer engineering from Korea University, Seoul, Korea, in February 2008. Currently he is working as a research engineer in LG Electronics, Korea.

Jun-Seong Kim received the BS degree in electronics engineering from Korea University, Seoul, Korea, in 2006. Currently, he is in the joint MS and PhD course at Korea University, Seoul, Korea. His research interests include image processing and multimedia communications.

Chang-Su Kim received the BS and MS degrees in control and instrumentation engineering from Seoul National University (SNU), Seoul, Korea, in 1994 and 1996, respectively. In 2000, he received the PhD degree in electrical engineering from SNU with a Distinguished Dissertation Award. From 2000 to 2001, he was a visiting scholar with the Signal and Image Processing Institute, University of Southern California, Los Angeles, and a consultant for InterVideo Inc., Los Angeles. From 2001 to 2003, he coordinated the 3D Data Compression Group in National Research Laboratory for 3D Visual Information Processing in SNU. From 2003 and 2005, he was an assistant professor in the Department of Information Engineering, Chinese University of Hong Kong. In Sept. 2005, he joined the Department of Electronics Engineering, Korea University, where he is now an associate professor. His research topics include image, video, 3D graphics processing, and multimedia communications. He has published more than 120 technical papers in international conferences and journals.