Disjunctive Normal Form of Weak Classifiers for Online Learning based Object Tracking

Zhu Teng and Dong-Joong Kang
Institute of Mechanical Engineering, Pusan National University
Busandaehak-ro 63beon-gil Geumjeong-gu, Busan, South Korea

Keywords: DNF of Weak Classifiers, Object Tracking, Online Learning.

Abstract: The use of a strong classifier that is combined by an ensemble of weak classifiers has been prevalent in tracking, classification etc. In the conventional ensemble tracking, one weak classifier selects a 1D feature, and the strong classifier is combined by a number of 1D weak classifiers. In this paper, we present a novel tracking algorithm where weak classifiers are 2D disjunctive normal form (DNF) of these 1D weak classifiers. The final strong classifier is then a linear combination of weak classifiers and 2D DNF cell classifiers. We treat tracking as a binary classification problem, and one full DNF can express any particular Boolean function; therefore 2D DNF classifiers have the capacity to represent more complex distributions than original weak classifiers. This can strengthen any original weak classifier. We implement the algorithm and run the experiments on several video sequences.

1 INTRODUCTION

Interest in motion analysis has recently increased in tandem with the development of enhanced motion analysis methodology and processing capabilities. Tracking entails following the motion of a smaller set of interest points or objects in video sequences, and is accordingly one of the most significant categories of motion analysis. Many applications of tracking (Avidan, 2004, Stauffer, 2000, etc.), including human face tracking, pedestrian tracking, and vehicle tracking, have been developed in accordance with the widespread use of surveillance. Taking tracking as a binary classification problem was first addressed in the mean-shift algorithm of (Comanciu, 2003), which trains a classifier to differentiate an object from the background. As encouraging results have been obtained (Parag, 2008, Tieu, 2000, Kalal, 2010, etc.), this approach has come into wide use. The classifier can be trained offline or online. The difference between offline learning and online learning is that offline learning requires the entire training set to be available at once, and sometimes it requires random access to the data, while online learning only involves one pass through the training data (Oza, 2001). Furthermore, offline learning methods have limited adaptability to variation of the objects. (Oza, 2001) and (Freund, 1995) present both the theoretical and experimental evidence that online boosting can achieve comparable performance to its offline counterparts. Our work concentrates on online boosting.

Online boosting has been studied by many researchers, and it is the most successful ensemble learning method. Shai Avidan proposed ensemble tracking, which combines a collection of weak classifiers into a single strong classifier, and treats tracking as a binary classification problem. A feature selection framework based on online boosting is introduced in (Grabner, 2006). An online semi-supervised boosting has been presented in (Grabner, 2008); it ameliorates the drifting problem in tracking applications by combining the decision of a given prior and an on-line classifier. (Stalder et al., 2009) further amalgamated a detector, recognizer, and tracker to track various objects. (Danielsson et al., 2011) used two derived weak classifiers to suppress combinations of weak classifiers whose outputs are anti-correlated on the target class. If a drifting problem occurs, it suggests that the error, which may be magnified, results in an incorrect decision of the object in object tracking, i.e., adapting to other objects. Though an error accumulation can also lead to a drifting problem, the fundamental reason for drifting is the erroneous estimation of the object,

Teng Z. and Kang D..
Disjunctive Normal Form of Weak Classifiers for Online Learning based Object Tracking.
DOI: 10.5220/0004240501380146
In *Proceedings of the International Conference on Computer Vision Theory and Applications* (VISAPP-2013), pages 138-146
ISBN: 978-989-8565-48-8

which is determined by the classifier when tracking is considered as a binary classification problem.

In this paper, we propose a 2D disjunctive normal form (DNF) of weak classifiers. The conventional weak classifier uses linear classifiers or stumps, which label samples just better than random guessing. Generally, this classifier takes the form of a threshold. A sample is tagged to an object category when the feature of the sample is larger or lower than the threshold. This conventional weak classifier is termed 1D weak classifier in our paper. The input data of the 2D DNF of weak classifiers are constituted by all the pairwise combinations of data utilized by all the 1D weak classifiers, and thus this approach is more accurate. As one full DNF can represent any particular Boolean function, the 2D DNF can express more difficult distributions than the conventional weak classifiers, and it also can be employed on top of any original weak classifier. To resolve the drifting problem, we combine it with a reset mechanism. On the one hand, the DNF can substantially decrease the error rate, which is the fundamental cause of drifting, and on the other hand, the reset mechanism suppresses error accumulation.

The contributions of this paper include: (i) the formulation of a novel type (DNF) of weak classifiers, and (ii) diversified features used in the tracking system, which is implemented by analyzing manifold features in the feature pool from the first frame of the video and determines the most appropriate features.

The reminder of this paper is organized as follows: Section 2 provides a brief introduction of AdaBoost. DNF tracking is illustrated in Section 3, along with definitions and applications of DNF classifiers. Section 4 presents the experiments and conclusions follow in Section 5.

2 ADABOOST AND ENSEMBLE TRACKING

To explain the basic notation, we will first briefly review AdaBoost (Freund, 1995). A strong classifier of AdaBoost is implemented by combining a set of weak classifiers. Many tracking algorithms are developed based on AdaBoost (Avidan, 2005). Generally, the algorithm is based on pixels, and the strong classifier determines if a pixel belongs to the object or not. It employs the addition and removal of weak classifiers to adapt to variation of the object appearance or background. The weak classifier used in (Avidan, 2005) is a linear classifier in a least-squares manner or other classifiers (such as stumps, perceptrons). Each pixel is represented by an 11D feature vector, which is created by a combination of the local orientation histogram and pixel colors. This feature vector can be computed easily, and is appropriate for object detection tasks (Levi, 2004).

Weak classifier:

Let $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ denote N examples and their labels, respectively, and $\mathbf{x}_i \in R^{11}$ and $y_i \in \{-1,+1\}$; the weak classifier can then be represented by

$$h_t(\mathbf{x}): R^{11} \to \{-1,+1\}, t \in [1,T] \quad (1)$$

where T is the number of weak classifiers.

Strong Classifier:

The strong classifier is defined as a linear combination of a collection of weak classifiers. It is given by *sign(H(**x**))* and

$$H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \quad (2)$$

$$\alpha_t = \frac{1}{2}\log\frac{1-err_t}{err_t} \quad (3)$$

$$err_t = \sum_{i=1}^{N} w_i \mid h_t(\mathbf{x}_i) - y_i \mid \quad (4)$$

where w_i is the weight of the i^{th} example, and weights are updated in the process of training weak classifiers (Eq. (5)).

$$w_i = w_i e^{(\alpha_t \mid h_t(\mathbf{x}_i) - y_i \mid)} \quad (5)$$

3 DNF TRACKING

In this session, we propose a novel tracking method that is based on the 2D DNF classifier. Session 3.1 describes the motivation for using the 2D DNF classifier rather than the 1D weak classifier in tracking, and defines the 2D DNF classifier. Procedures for tracking based on 2D DNF classifiers are illustrated in Session 3.2.

3.1 DNF Classifier

The proposed 2D DNF classifier is first motivated by the drifting problem in ensemble tracking. The drifting problem entails two important aspects: the fundamental cause of drifting is the misclassification

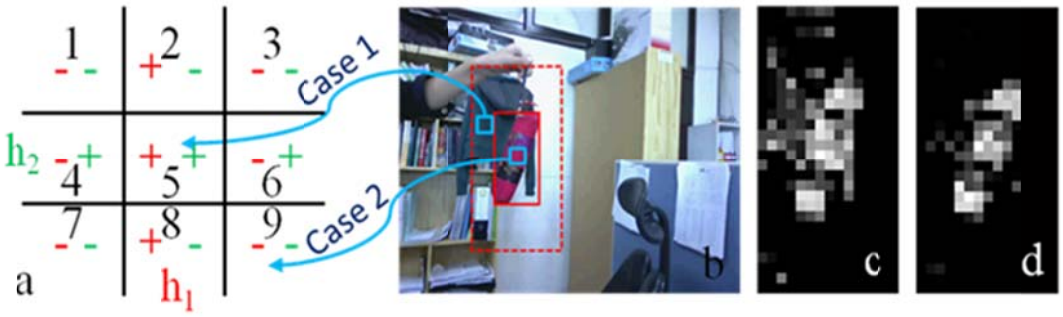

Figure 1: Stimulus example. (a) 2D feature space. (b) The first frame with the object in the solid red rectangular. Confidence map of the second frame (c) and the fifth frame (d).

rate of weak classifiers, and the other is the remedial action if drifting takes place. Even though the final strong classifier is a linear combination of weak classifiers, there are still many distributions that linear combinations cannot present.

Take a 2D feature space as an example (O. Danielsson, 2011). Assume there are two weak classifiers, $h_1(\mathbf{x})$ and $h_2(\mathbf{x})$, each classifying the feature space into positive and negative regions as shown in Fig. 1 (a). A region with a red plus sign suggests that classifier $h_1(\mathbf{x})$ categorizes it as a positive region, and a red minus sign stands for a negative region that is labelled by classifier $h_1(\mathbf{x})$. We describe the green plus sign and green minus sign of classifier $h_2(\mathbf{x})$ similarly. As a result, in Case 1 the strong classifier cannot classify the background patch correctly (always '+'), and in Case 2 the object patch cannot be correctly labelled (always '-'). We see that some regions are always classified as '+' (such as region 5), and some regions are always classified as '-' (such as region 9). The linear combination of weak classifiers (or the strong classifier) cannot overcome this problem.

2D DNF cell classifier:

Let $\{\mathbf{p}_f, \mathbf{y}\}$ denote examples of the f^{th} cell and their labels, respectively, where each element of $\mathbf{y}$ should be the value *-1* or *+1*, $\mathbf{p}_f = [\mathbf{d}_{h_i}; \mathbf{d}_{h_j}]_{2\times N}$, N is the number of examples, and $\mathbf{d}_{h_i}$ and $\mathbf{d}_{h_j}$ are the feature data that weak classifiers h_i and h_j have employed in classification. The plane $d_{h_i} - d_{h_j}$ is quantified into $m \times m$ bins, denoted by $\mathbf{b}_{ij}, 1 \le i, j \le m$. The 2D DNF cell classifier is defined as:

$$h_{2Dcf}(\mathbf{p}_f) : R^2 \to \{-1, +1\} \tag{6}$$

For each column vector $\mathbf{p}_{fi}$ in $\mathbf{p}_f$, the specific mapping relationship is presented in Eq. (7).

$$h_{2Dcf}(\mathbf{p}_{fi}) = \begin{cases} 1, & \mathbf{p}_{fi} \in \bigcup_{1\le i,j\le m} Cb_{ij} \\ -1, & otherwise \end{cases} \tag{7}$$

$$Cb_{ij} = \begin{cases} b_{ij}, & |\{y_k | \mathbf{p}_{fk} \in b_{ij} \wedge y_k = 1\}| > |\{y_k | \mathbf{p}_{fk} \in b_{ij} \wedge y_k = -1\}|, k \in [1, N] \\ \varnothing, & otherwise \end{cases} \tag{8}$$

$|.|$ indicates the cardinality of a set in Eq. (8). Cb_{ij} represents the bin b_{ij} if there are more positive examples than the negative examples in that bin, otherwise it is a null set. $\bigcup_{1\le i,j\le m} Cb_{ij}$ is the union of all bins that have more positive examples than the negative examples in each bin. An example is classified as positive if it enters into any bin of this union.

As a simple example, consider R channel of the image as a 1D feature and 1D of EOH (Levi, K., 2004) related to edge as another 1D feature. If the total number of red pixels in the background is larger than that in the object, then red pixels in the object might be recognized as background when training by 1D feature of R channel (weak classifier). If the red pixels of background do not have the same EOH feature as the object, 2D cell classifier can differentiate them, and then the red pixels of the object are possible to be recognized as object.

2D DNF classifier:

The 2D DNF classifier is defined as a linear combination of a set of 2D DNF cell classifiers, and the number of 2D DNF cell classifiers is denoted by *M*.

$$H_{2D}(\mathbf{x}) = \sum_{f=1}^{M} \alpha_{2Df} h_{2Dcf}(\mathbf{p}_f) \quad (9)$$

Let $S = \{\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_w\}$ be the set of feature data that weak classifiers have used; then $\{\mathbf{p}_f\}$ is a set of all the pairwise combinations of the set *S*, that is, $\mathbf{p}_f$ is a subset of two distinct elements of *S* (as shown in Eq. (10)).

$$\{\mathbf{p}_f\} = \bigcup_{1 \le i, j \le w, i < j} \{\mathbf{d}_i, \mathbf{d}_j\} \quad (10)$$

w is the row number of feature data that weak classifiers have employed in Eq. (10), and it is no larger than the dimension of the feature space.

3.2 2D DNF Classifiers for Tracking

To start tracking, feature data are first extracted. Diverse kinds of features are obtained from the first frame of the video sequence, and they are employed to train the new weak and DNF cell classifiers. The best kind of feature according to the performance of these features on the first frame is then selected by the feature pre-selector. Classifiers are trained in initialization, and are constantly updated in the following frames. We use an ensemble of weak and DNF classifiers to determine whether a patch belongs to the object or not, and a confidence map is also constructed during this process. The peak of the map, which is achieved by the integral image (P. Viola, 2001) of the confidence map, is believed to be the new object position. The feature data of the new position of the object are used to update classifiers.

3.2.1 Feature Pre-selector

In order to track different objects, the most suitable features to employ are not always the same, and feature selection techniques have been researched by many researchers (see Ref (R. Collins, 2005) as an example). In order to apply the most appropriate features in diversified tracking missions, the feature pre-selector is constituted. It is a product of the compromise of the amount of computation and the adaptability of different objects to be tracked. All kinds of the features are calculated from the first frame of the video sequence. Features of a fixed number of patches used for learning are randomly selected, and the performance of classifiers for each kind of feature is assessed on other randomly selected patches for the first frame. The feature pre-selector chooses the feature with the best performance. After the type of feature is determined, the remaining frames will only calculate this kind of feature. Therefore, the time required for calculating features is reduced as only the pre-selected feature is employed once tracking has commenced. The features used in this work include the local binary pattern (LBP) (T. Ahonen, 2004), Haar feature (P. Viola, 2001, Papageorgiou, 1998), and local edge orientation histograms (EOH) (Levi, K., 2004). All these features are extracted based on patches and are combined with the average R, G, and B values in each channel of the patches.

3.2.2 Outlier Elimination

Outliers in our work are defined as patches in the bounding box of the object but do not belong to the object. Outliers can affect weak and DNF classifiers in the processes of training and updating.

In the initialization step when training classifiers, even though the object is given by a bounding box, patches in this bounding box do not always belong to the object to be tracked, because the object is not always a shape of rectangle. This kind of outliers is represented as minority points in the bins of feature space because outliers are minority compared to the majority features of the object in the bounding box. To apply it to our work, Eq. (8) is changed to Eq. (11) in the real implementation as shown below. The parameter *r* should be a positive integer (we set it to 5 in our experiments).

$$Cb_{ij} = \begin{cases} b_{ij} & \left| \{y_k | \mathbf{p}_{fk} \in b_{ij} \wedge y_k = 1\} \right| - \left| \{y_k | \mathbf{p}_{fk} \in b_{ij} \wedge y_k = -1\} \right| > r, k \in [1, N] \\ \varnothing & otherwise \end{cases} \quad (11)$$

We first attempted to use the labeled data in the last frame to update classifiers, i.e. semi-supervised learning, which lends adaptability to the system; however, if mistakenly estimated data are used, the system can easily drift. In other words, when classifiers are updated, patches in the bounding box are labelled as positive example (Eq. 12, pa_i represents the i^{th} patch), whereas in fact they should be labeled as negative examples.

$$y_i = \begin{cases} +1 & pa_i \text{ is in the object rectangle} \\ -1 & otherwise \end{cases} \quad (12)$$

If we do not reject these patches, they will be trained as the object, which may lead to drifting. Employing it in our algorithm, patches in the bounding box that have a relatively larger confidence can be labelled as positive (Eq. (13)).

$$y_i = \begin{cases} +1 & (pa_i \in object\ rectangle) \wedge (conf(pa_i) > 0.5) \\ -1 & otherwise \end{cases} \quad (13)$$

3.2.3 Specific DNF Algorithm of Tracking

Algorithm 1*: DNF algorithm for tracking*
Input: a video sequence with n frames;
a bounding box for the object in the first frame.
Output: a bounding box of the object for the remaining frames.

Initialization (for the First Frame)*:*
(1) Extract all types of features from the first frame $\{\mathbf{x}_f, \mathbf{y}_f, f \in [1, F]\}$*, where F is the total number of types of features. The number of positive and negative patches used for training is fixed, and these patches are randomly selected.*

(2) Train weak classifiers and 2D DNF classifiers for each type of feature. Randomly select patches from the first frame, extract features of these patches as test examples, and the feature with the minimum error is chosen for use in the following frame.

(3) Set the state of tracking as FOUND, and save initial classifiers and data.
For a New Frame*:*

(1) Draw the pre-selected feature of all the patches from the background of the current frame. Generally, the background is defined as twice the size of the object, while the detected region is spread to the whole frame in the case of losing the object.

(2) Examine all the patches with the combination of weak classifiers and 2D DNF cell classifiers $H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) + \sum_{f=1}^{M} \alpha_{2Df} h_{2Dcf}(\mathbf{p}_f)$ *, and the confidence map is created.*

$$confidence = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) + \sum_{f=1}^{M} \alpha_{2Df} h_{2Dcf}(\mathbf{p}_f)$$

(3) Obtain the object position and the current confidence from the integral image of the confidence map. If the current confidence is not larger than a threshold TH1, the state of tracking is determined as LOST. The classifiers are restored to their initial states, and the detected region is spread to the whole frame.

(4) If the current frame is under the LOST state and the current confidence is larger than a threshold TH2, the state of tracking is reinstated to the FOUND state.

(5) If the current frame is under the FOUND state, update classifiers.

In the update step, the positive data for updating are comprised of the labeled data from the last frame and the initial positive data. The updating of weak classifiers is the same with (Avidan, 2005), and as the weak classifiers update, the data for the DNF classifiers are updated and the DNF classifiers are renovated.

4 EXPERIMENTS

In this section, we implement the proposed algorithm in Matlab, evaluate it on several video sequences, and compare its performance with that of three other tracking methods. We also use sequences of PROST dataset and the evaluation method provided by (Santner, 2010) to demonstrate the performance of our algorithm. Furthermore, the performance of the DNF cell classifier is weighed against that of a weak classifier in Section 4.1, the performance comparison of DNF classifier and strong classifier is presented in Section 4.2, and the effects of exclusion of outliers are illustrated in Section 4.3. All of the experiments are executed on an Intel(R) i5 2.80GHz desktop computer.

4.1 2D DNF Cell Classifier VS Weak Classifier

This experiment is carried out to evaluate the performance of the DNF cell classifier, the performance of which is also compared with that of weak classifiers (Avidan, 2005). The data used in this experiment are the 9th dimension and 10th dimension data of EOH feature and are normalized to the range [0, 1]. The feature is calculated based on patches, the radius of which is set to 5. Classifiers are trained on the first frame of the video sequence and updated in the following frames; Fig. 2 shows the results of the fifth frame. Features of all the patches in the fifth frame are extracted. Classifiers are then applied to these features and the patches are classified to the object category or background category. Each point in Fig. 2 represents a patch in the image, where a red plus sign indicates an object patch and a green point denotes a background patch. We show two situations of the ground truth. Only object patches are set to positive data (Fig. 2 (a)) for the first situation, that is, with ideal outliers excluded. For the other situation, patches in the bounding box are all put to the positive data set (as shown in Fig. 2 (b)). It is obvious that the red plus signs in the black ellipses of Fig. 2 (b) are outliers. For instance, patches of the background coat (green

color) in the solid red rectangle (object bounding box) in Fig. 1 (b) are this kind of outlier. We can see that the performance of the DNF cell classifier (Fig. 2 (e)) is much better than that of weak classifiers (Figs. 2 (c, d)) even though it is slightly influenced by the outliers in the object bounding box. If the performance of the cell classifiers is good, we can expect the final DNF classifier will be better.

4.2 2d DNF Classifier Vs Strong Classifier

Fig. 2 (f) shows the error rates of three classifiers to demonstrate to what extent 2D DNF classifier can improve the performance, compared with the strong classifier. As the experiment is to compare the classification capability of these three classifiers, no updates or other techniques are used (such as outlier elimination). We train the three classifiers on the first frame, and test on more than 260 other frames (the video sequence used here is "car", see also Fig. 3 (a)). For each frame, features of all the patches are calculated (EOH feature is employed), and the error rate is defined as the number of correctly classified patches divided by the whole number of patches. Furthermore, we add the only-DNF classifier, which is only a linear combination of DNF cell classifiers. The combined classifier in the Fig. 2 (f) is the classifier used in Algorithm 1, which is a linear combination of weak classifiers and 2D DNF cell classifiers. It is clear that the combined classifier has the best classification capability compared to the other two classifiers.

4.3 Outlier Elimination Experiment

The goal of this experiment is to view the effects of outlier elimination (shown in Fig. 1). In the initialization step, positive training data that are obtained from the object bounding box of the first frame (Fig. 1(b)) include data that do not belong to the object, and if these data are not rejected in the updating process, the outliers will be trained in the same manner as the object, which leads to the drifting of the tracker. In each bin of feature space, there are more patches from the object than from outliers, even though these outliers are labelled as the object, many of them cannot win over the object data (Eq. (11)). Furthermore, most of the winning outliers can be restrained in the updating procedure, as patches with lower confidence are not updated to the next frame. As shown in Fig. 1(c), patches from the green coat in the background are initially trained as the object, but are soon restricted in the following frames (Fig. 1(d)) as the outlier exclusion takes effect.

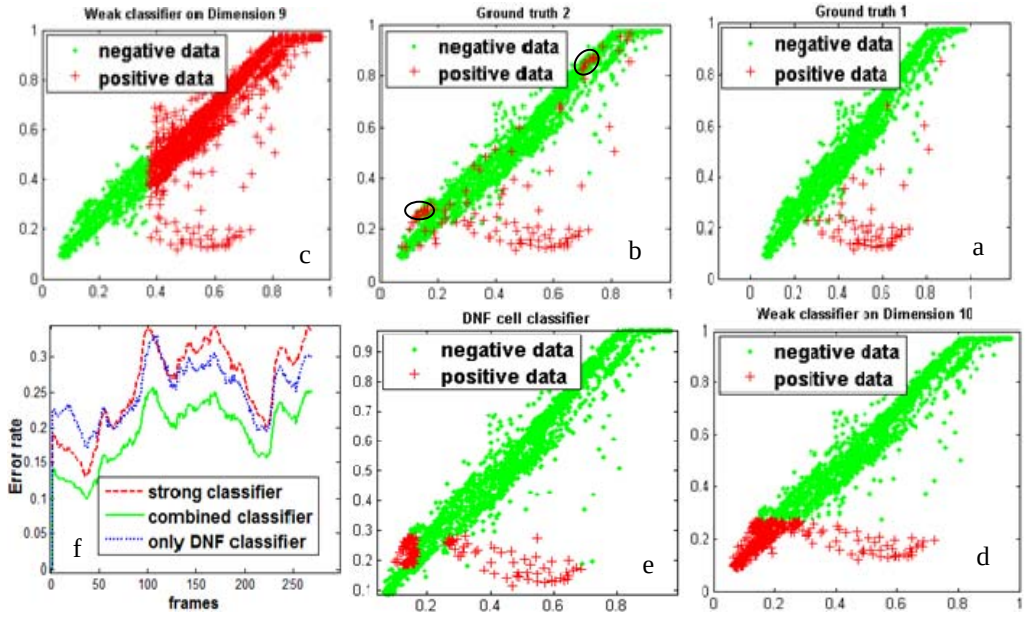

Figure 2: Experiments for comparisons of weak classifiers and DNF cell classifiers. a) Ground truth with only patches from the object set as positive data; b) Ground truth with patches in the object bounding box set as positive data; c) classifying results of weak classifier 1; d) classifying results of weak classifier 2; e) classifying results of DNF cell classifier; f) comparative results of strong classifier and DNF classifier.

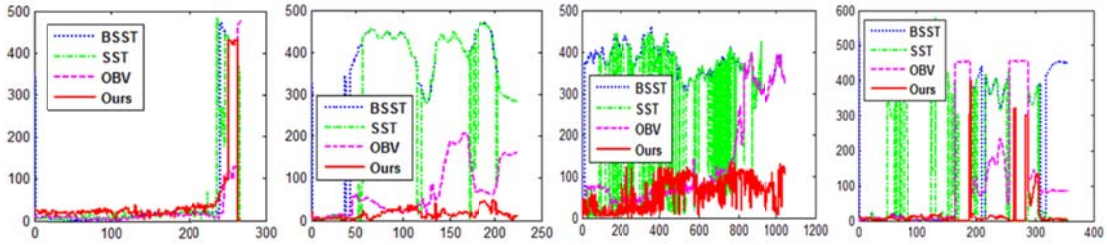

Figure 3: Center differences (in pixels) between the ground truth and the tracking results. From left to right: a) car, b) pencil case, c) pedestrian, d) cup setting as the tracking object.

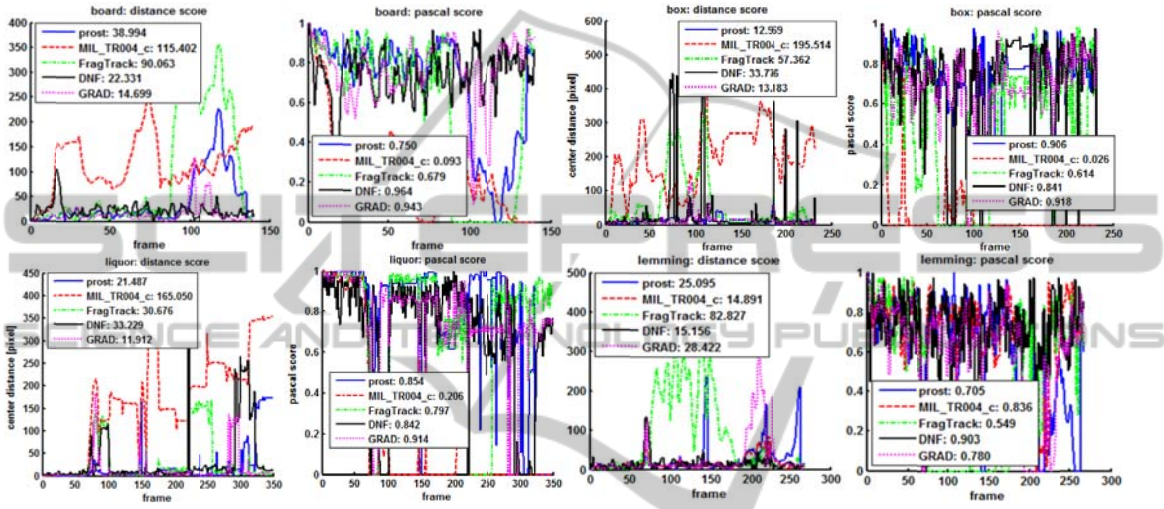

Figure 4: Comparative results of DNF and other four methods in the video sequences provided in Ref (Jakob Santner, 2010). Distance score: the mean center location error in pixels. Pascal score: calculated by Eq.(14).

4.4 Experiments on Video Sequences

The proposed algorithm can track a variety of objects. Our algorithm and three other methods (The source codes of all the three methods are available at http://www.vision.ee.ethz.ch/boostingTrackers/index.htm) are executed to track different objects in several video sequences. The other three methods are BSST of (S. Stalder, 2009), SST of (H. Grabner, 2008), and OBV of (H. Grabner, 2006). The size of all the video sequences is 640*480 and the object is provided in the first frame manually by a bounding box. The tracking results of the four video sequences are shown in Fig. 3, the vertical axis of which is the Euclidean distance between the center of the bounding box of the detected object and the center of the ground truth bounding box. The ground truth positions of objects are acquired manually frame by frame. Parts of the frame shots are shown in Fig. 5.

The first video sequence is a car running on a road. All four methods tracked the car well at first, but the performances varied as the car gradually disappeared from the clip. BSST and SST lost the object at the frame where roughly 90% of the object was included, OBV mistakenly detected another place as the object (see Fig. 5 (1e) and (1f)), and our method lost the object when 50% of the object disappeared from the video sequence. From Fig. 3 (a) we can see that the curves of BSST, SST, and our method converge to zero, as these three methods did not detect the object. It is the same as the ground truth, because finally the car disappeared from the clip. Meanwhile, OBV mistakenly tracked another place as object, resulting in a large center difference. In the second clip, a pencil case was tracked. The bounding box of the pencil case included many background pixels when it is lifted up, and this makes tracking easily drifted (as shown in Fig. 5(2e)), especially in the case where the background of the object bounding box is the same. A pedestrian was tracked in the third video sequence. In this experiment, BSST lost the object at about the 20th frame and was not able to re-track the object, SST alternated between detecting the pedestrian and losing the pedestrian, and OBV initially showed good performance but detected the wrong object at

about the 700th frame (as shown in Fig. 5(3e) and (3f)) and could not recover the detection of the pedestrian thereafter (see Fig. 3(c)). Our method provides a relatively good performance, but the center difference is somewhat large (the other three methods suffer the same problem). The reason for this is that the pedestrian in this clip was sometimes standing near the camera and appeared larger than that in the initial frame but the size of the object bounding box in our method is fixed during the tracking process. The object of the fourth clip is a cup. The cup disappeared twice in the video sequence. The first disappearance was at about the 165th frame, and OBV lost the object from this frame on (see Fig. 3(d)). In the case of the SST method, the object was lost and recovered a number of times. BSST provided relatively stable tracking but it failed to track the object between two disappearances. Adaptation to other objects occurred occasionally in our method as well. However, this was remedied quickly, which is manifested as sharp peaks in Fig. 3(d).

Besides these video sequences, we also testify our method on the PROST dataset, the video sequences in which were newly created by the authors of (Jakob Santner, 2010) (The video sequences and the code of the evaluation method are available at http://gpu4vision.icg.tugraz.at/index.php?content=subsites/prost/prost.php), and the two evaluation methods shown in Fig. 4 are also provided by (Jakob Santner, 2010). The first evaluation is the distance score that represents the mean center location error in pixels. The second evaluation method is PASCAL score based on PASCAL challenge (M. Everingham, 2009). A frame is determined as a corrected tracked frame if the overlap score of the frame proceeds 0.5. The overlap score is calculated by Eq. (14), where BB_D denotes the detected bounding box and BB_{GT} represents the ground truth bounding box. Each point on the PASCAL score curve of Fig. 4 is the overlap score for each frame, and the number in the graph legend of PASCAL score figure represents a percentage of correctly tracked frames for a sequence.

$$score = \frac{area(BB_D \cap BB_{GT})}{area(BB_D \cup BB_{GT})} \quad (14)$$

The benchmarked methods of Fig. 4 involves the

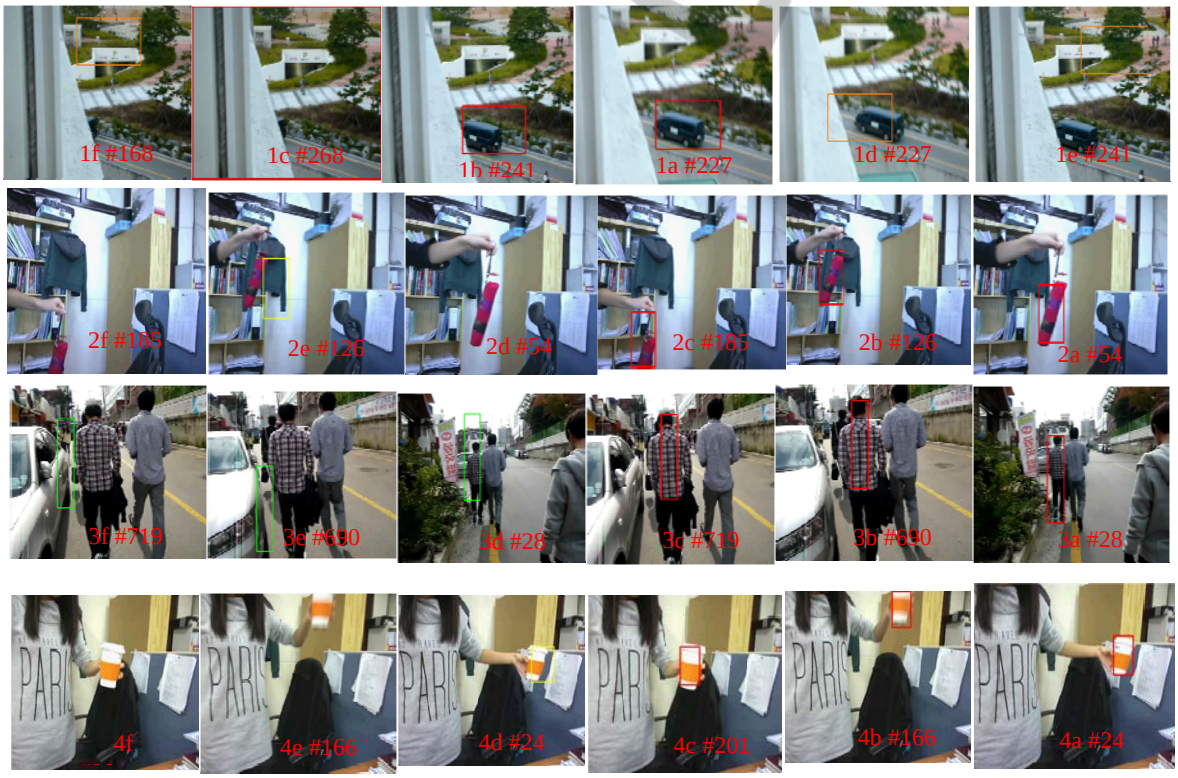

Figure 5: Parts of frames of experimental results on video sequences. Processing methods: 1d-1f: SST, 2d-2f: BSST, 3d-3f: OBV, 4d-4f: BSST, 1a-1c,2a-2c,3a-3c,4a-4c: our method.

methods of PROST (Jakob Santner, 2010), MIL_TR004_c (B. Babenko, 2009), FragTrack (A. Adam, 2006), and GRAD (Klein, 2011). It shows that our method achieves a best performance in sequences of board and lemming, and a slightly less good performance than PROST in sequences of liquor and box. An average PASCAL score of our method over the four sequences is 88.75%, which is much better than the average of 80.375% for PROST method.

5 CONCLUSIONS

This paper described a novel tracking method based on a 2D DNF of weak classifiers. The data of the DNF cell classifiers are constituted by pairwise combinations of the data of weak classifiers, and therefore the DNF can be utilized on top of any weak classifiers. The image patch is determined to belong to the object category or the background category by an ensemble of weak classifiers and DNF cell classifiers. The experiments demonstrate that our method provides a good performance compared to other methods but sometimes the center difference is somewhat large due to the unvaried object bounding box. For better tracking, we will continue the present line of research with a scalable object bounding box in the future.

ACKNOWLEDGEMENTS

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2009-0090165, 2011-0017228).

REFERENCES

Avidan S., 2004. Support Vector Tracking. In *IEEE Trans. On Pattern Analysis and Machine Intelligence*.

Stauffer, C. and E. Grimson, 2000. Learning Patterns of Activity Using Real-Time Tracking. In *PAMI, 22*(8):747-757.

S. Avidan, 2005. Ensemble tracking. In *Proc. CVPR, volume 2*, pages 494–501.

H. Grabner and H. Bischof, 2006. On-line boosting and vision. In *Proc. CVPR, volume 1*, pages 260–267.

H. Grabner, C. Leistner, and H. Bischof, 2008. Semi-supervised on-line boosting for robust tracking. In *Proc. ECCV*.

S. Stalder, H. Grabner, and L. van Gool, 2009. Beyond Semi-Supervised Tracking: Tracking Should Be as Simple as Detection, But Not Simpler than Recognition. In *Proc. Workshop Online Learning in Computer Vision*.

O. Danielsson, B. Rasolzadeh, and S.Carlsson, 2011. Gated Classifiers: Boosting under High Intra-Class Variation. In *Proc. CVPR*.

N. Oza and S. Russell, 2001. Online bagging and boosting. In *Proc. Artificial Intelligence and Statistics*, pages 105–112.

Freund, Y. Schapire, R. E, 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt* 95, pp 23-37.

Comanciu, D., Visvanathan R., Meer. P, 2003. Kernel-Based Object Tracking. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 25:5, pp 564-575.

Levi, K., Weiss, Y, 2004. Learning Object Detection from a Small Number of Examples: The Importance of Good Features. In *IEEE Conf. on Computer Vision and Pattern Recognition*.

P. Viola and M. Jones, 2001. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR, volume I*, pages 511–518.

Papageorgiou, Oren and Poggio, 1998. A general framework for object detection. In *International Conference on Computer Vision*.

T. Ahonen, A. Hadid, and M. Pietikäinen, 2004. Face Recognition with Local Binary Patterns. In *Proc. Eighth European Conf. Computer Vision*, pp. 469-481.

T. Parag, F. Porikli, and A. Elgammal, 2008. Boosting adaptive linear weak classifiers for online learning and tracking. In *Proc. CVPR*.

K. Tieu and P. Viola, 2000. Boosting image retrieval. In *Proc. CVPR*, pages 228–235.

Z. Kalal, J. Matas, and K. Mikolajczyk, 2010. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. In *Proc. CVPR*.

Jakob Santner, Christian Leistner, Amir Sa_ari, Thomas Pock, and Horst Bischof, 2010. Prost: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 723-730, 13-18.

B. Babenko, M.-H. Yang, and S. Belongie, 2009. Visual Tracking with Online Multiple Instance Learning. In *CVPR*.

Klein, Cremers, 2011. Boosting Scalable Gradient Features for Adaptive Real-Time Tracking. In *Int. Conf. on Robotics and Automation (ICRA)*.

A. Adam, E. Rivlin, and I. Shimshoni, 2006. Robust fragments based tracking using the integral histogram. In *CVPR*.

R. Collins, Y. Liu, and M. Leordeanu, 2005. Online selection of discriminative tracking features. In *PAMI, 27*(10):1631–1643.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, 2009. The Pascal Visual Object Classes (VOC) Challenge. In *Int. J. Comput. Vision, 88(2):*303–308.