國 立 交 通 大 學

電控工程研究所

碩士論文

利用景深資訊降低背景干擾以減緩飄移問題

Eliminating the Drifting Problem with Background Interference Reduction using Depth Information

研 究 生：黃錦銘

Student: Kingming Huang

指導教授：黃育綸　博士

Advisor: Dr. Yu-Lun Huang

中華民國一百年六月

June, 2011

利用景深資訊降低背景干擾以減緩飄移問題

Eliminating the Drifting Problem with Background Interference Reduction using Depth Information

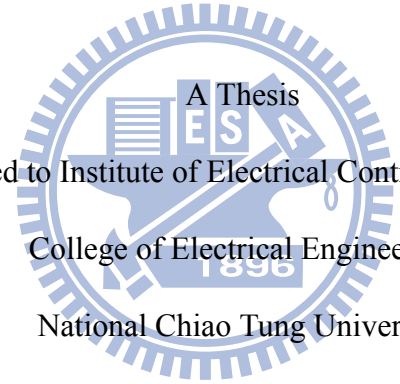
研 究 生：黃錦銘 Student: Kingming Huang

指導教授：黃育綸　博士 Advisor: Dr. Yu-Lun Huang

國 立 交 通 大 學

電控工程研究所

碩士論文



A Thesis

Submitted to Institute of Electrical Control Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfill of the Requirements

for the Degree of

Master

in

Institute of Electrical Control Engineering

June, 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年六月

利用景深資訊降低背景干擾以減緩飄移問題

學生： 黃錦銘　　指導教授：黃育綸　博士

國 立 交 通 大 學電控工程研究所（研究所）碩士班

摘　　要

使用適應性物體模型是一種的物體追蹤方法。 這種追蹤法具有演算法簡單與執行快速的優點，但也容易因為背景的干擾問題，出現飄移（Drifting）問題，而影響追蹤結果的正確性。 飄移問題的發生主因來自於 1) 物體的適應能力，以及 2) 背景的干擾。 在這篇論文中，我們以 Online Boosting for Tracking (OBT) 演算法為基礎，引入了景深、多尺度的追蹤器和動態更新的追蹤器生命值等資訊，設計了一套新的物體追蹤演算法，稱為 Enhanced OBT（簡稱 EOBT）。 在 EOBT 中，景深資訊可用來濾除背景、多尺度的追蹤器可以改善追蹤的準確度，而動態更新的追蹤器生命值則可用以判斷物體是否被短暫的遮蔽，進而降低追蹤器因物體被短暫遮蔽所造成的準確度影響。 此外，由於現有的準確度評估方法無法完全反映出追錯目標物體的問題，在本論文中，我們另外提出了新的評估方法，設計新的比率（Ratio in Object 及 Ratio in Tracker）來評估追蹤的準確度。 其中，Ratio In Object 反映了有多少比率的物體被成功地追蹤到；而 Ratio In Tracker 則反映出待追物體落在追蹤器內的面積比率。 我們也設計了不同的實驗，證明本論文所提出之 EOBT 演算法能成功地減緩飄移問題，並提高物體追蹤的準確度。

Eliminating the Drifting Problem with Background Interference Reduction using Depth Information

Student: Kingming Huang Advisor: Dr. Yu-Lun Huang

Institute of Electrical Control Engineering

National Chiao Tung University

Abstract

Recently, tracking using adaptive appearance models is popular. Tracking algorithms adopting an adaptive appearance model are simple and fast, but suffer from drifting problems caused by background interference. The drifting problem, resulting in inaccuracy, comes from the accumulation of slight labeling errors occur in updating model in each tracking iteration. Taking online boosting for tracking (OBT) as the basis, we introduce depth, multiple scales and lifetimer to our algorithm (named Enhanced OBT; also abbreviate to EOBT) and eliminate drifting problems induced by background interference. In EOBT, depth can be used to filter out the background data, the tracker with multiple scales can be used to improve the accuracy, and dynamically adjusted lifetimer can be used to determine whether the object is temporarily occluded. Since conventional evaluation method of accuracy may derive a high accuracy when an algorithm tracks a wrong target, we additionally design two ratios (`Ratio in Object' and `Ratio in Tracker') to avoid such a problem and precisely evaluate the accuracy. In our method, `Ratio in Object' shows the percentage of an object caught by a tracker, while the `Ratio in Tracker' reflects the percentage of a tracker occupied by the object to be tracked. In this thesis, we conduct several experiments to show that EOBT can effectively reduce drifting problems and improve the accuracy of object tracking.

誌謝

碩論能夠完成要感謝很多人的幫助，首先感謝指導教授黃育綸博士在這碩士兩年的時間裡，給予相當自由的研究空間，並於關鍵時刻引導使我不至於走偏方向；當初若不是教授給予自由的研究空間，可能也無法投注夠足夠的熱情。

另外，還要感謝實驗室的同儕不僅在課業方面，也使我在這段兩年的研究生活中逐漸成長茁壯。依輩分列出分別是博班的三位學長姐蔡欣宜學姐、曾勁源學長以及陳柏廷學長，非常感謝曾勁源學長在研究上的指導；感謝蔡欣宜學姐和陳柏廷學長在論文、口試以及計畫方面的幫助。另外感謝甄元斌學長、黃啟彥學長、吳嘉稘學長、王雅萱學姐、吳思穎學姐、彭博群學長等學長姐；同屆的好友黃奕奇、許鴻生、黃晉澤以及和我一同從電工進入電控所的僑生鄭偉強；還有學弟妹們包括賴鈺婷、葉書宏、李勇叡、陳玟媗等人。實驗室的研究生生活因為有了你們，變得更加多采多姿。

最後，感謝我的家人們這些年來在背後默默的支持及鼓勵。感謝父母總是在做決定時給予我絕對的自由，但又不忘在該注意時提醒我。

碩士論文的完成總是需要感謝許多人的幫助，而這些感謝也是筆墨難以形容、難以列舉的。感謝在這段期間曾經幫助我的所有人，謝謝你們。

Contents

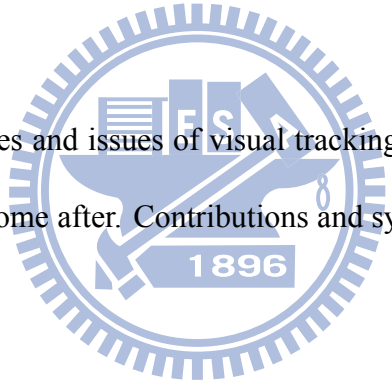
List of Figures

Chapter 1

Introduction

Visual tracking, or object tracking, has been studied for decades. The object for visual tracking is to continuously label objects of interest in video sequences. This is a prior step for further image processes. Lots of applications, like video indexing, human-computer interaction (HCI), traffic monitoring, augmented reality (AR), demand executions of visual tracking. For example, automated surveillance needs information about suspicious human motion. To acquire those information, human tracking shell be performed first, and then traces can be recorded for further analysis.

In this chapter, challenges and issues of visual tracking are discussed first. Recent developments of object tracking come after. Contributions and synopsis of this paper are mentioned at the end of this chapter.



1.1 Challenges

Tracking is continuously finding region of interest (ROI) in frame sequences. The most intuitive tracking method is recording every pixel value of ROI at first frame. Then, tracking is implemented as finding the most similar region as records in the succeeding frames. However, in frame sequences, there are lots of variations need to be took into consideration. For example, illumination, shape variations or occlusions. These variations make object tracking more difficult than imagination. Here, several challenges are summarized as follows [1].

Loss of Information

Images are formulated by projecting 3D real world on 2D sensors. Useful information like spatial structures has been destroyed.

Noise

Noises occur from different sources, from hardware to software. From signal processing viewpoint, when and where noises occur are unpredictable. To design a noise-sustained system is challenging.

Complex Object Motion

Object motions vary diversely. Thought kinematics theorems can be used, object motion in 3D environment is hard to predict. Some researchers have made use of this prediction as context information [2]. Using this method, the false positive rate of tracking accuracy decreases to an extent. However, computational cost should pay at the same time.

Object Body

Object could be coarsely categorized in rigid, non-rigid or articulated body. Their properties are different from objects to objects, and the borderlines between them are vague. Especially, articulating object has been independently addressed in recent research [3].

Occlusions

Due to loss of 3D information, partial or entire occlusions come up frequently. Recently, some researchers use adaptive appearance models as a tracking method. Adaptive appearance model continuously learn new appearances, which has the ability to conquer variations like illumination. Nevertheless, when occlusions happen, adaptive methods learn wrong appearances. This problem is coined as template update problem, or drifting, which is addressed in the rest of this thesis.

Complex Object Shapes

Object shapes cause problems. At first look, complex object shapes are hard to describe. Usually, primitive geometry shapes are used to stand for objects, but in the meantime, background is introduced. The drifting problem mentioned earlier also occurs in this situation.

Scene Illumination Changes

Illumination changes cause problems since object appearance learnt from previous frame is different from now. Previous object appearance can not be completely trusted. Several tracking methods have been proposed to solve this problem.

Processing Time Requirements

Though usually real-time, 20 frames per second (fps), is demanded, different situations change this requirement. For example, in normal video surveillances, 7 fps is fast enough, which eases time requirement. Trade-off between accuracy and processing time should be taken into account.

1.2 Issues of Existing Tracking Methods

Challenges of visual tracking have been discussed in previous section. This section focuses on issues of tracking methods. Each method has its applicability. For example, Kanade-Lucas-Tomasi (KLT) tracker [4] is properly used in augmented reality (AR) field, because object motions could be detected precisely. However, for a 24 hour surveillance system, illumination variation is one of main concerns. Cannons [5] and Yilmaz [1] have discussed different tracking methods using their own taxonomy. Here these methods are classified in functional aspects. First, the ways these methods represent ROI are described, with or without a priori model. Also,

their pros and cons are mentioned. Finally, comparisons on tracking methods and summary are at the end of this section.

1.2.1 Object Representations

First of all, object representations are tightly related to tracking methods and their applications. Once an object representation is adopted, the applicability has also been fixed. Yilmaz [1] has made an extensive survey on his paper. However, for clarity, we adopt Cannons' categorization [5]. He classifies these object representations into three main categories, points, edges and lines, and regions. These categories are introduced respectively as follows.

Points

Points has been an excellent object representation since Harris interest point detector was proposed [6]. Nowadays, local invariant point feature, like SIFT [7] and SURF [8] ,has gained lots of attention. Point representation is popularly used in Augmented Reality and other applications, because it possesses object orientations and can be computed very fast. Point representation is a simple method to represent objects. However, due to this simplified representation, lots of objects would have similar representations. In this situation, tracker might get confused and the odd of false positive is elevated. In reality, point representation does not precisely learn objects. All information they held are points. Tougher jobs like object recognition, point representation seems not applicable to.

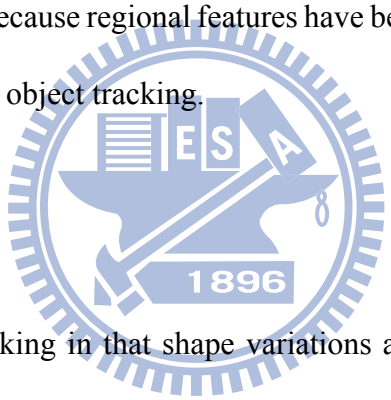
Edges and Lines

Edge features are used in many tracking systems. An edge is defined as pixels lie on the boundary between regions. Usually edges are grouped into lines, and an easy detection using filter banks can be applied. Cannon has addressed three categories of line detectors, including similarity according to two points, two parallel edges and 3D surface. Com-

monly used method is the famous Hough transform. In fact, due to discard too much potential information, tracking using edges or lines is getting less attention.

Regions

Recently, using regional features for tracking becomes popular. There are two types of regional descriptors, color histograms and histograms of gradients. Color histogram methods are invariant to translation and rotation. However, the spatial information collapses at the same time. Also, depending on color, illumination variations severely damage color histogram. On the other hand, histogram of gradients (HOGs) is robust to illumination variations, but easily affected by cluttered background. Therefore, hybrid methods have also been addressed. Because regional features have better ability to describe objects, they are extensively used in object tracking.



1.2.2 Object Models

Object models help tracking in that shape variations are handled. Using object models, object shape in three dimension space is revealed and predictable. In this way, more information can be gathered using kinematics. However, model construction is the bottleneck. If every object model is constructed before tracking, applicability is consequently reduced. In practical applications, human tracking with models is the most common usage, like Microsoft®XBOX 360. It is a trade off between tracking accuracy and applicability.

1.2.3 Tracking Methods

In Cannon's survey [5], four categories of tracking methods are classified. They are tracking using discrete features, tracking with contours, region-based trackers and combined trackers. In tracking using discrete features, analytic methods are used to match inter-frame relations.

Tracking with contours use methods like Snake [9] or level sets [10]. Region-based trackers and combined trackers are recently become popular. In region-based trackers, Cannon has addressed blob trackers, pixel-wise template tracking, kernel histogram methods. These methods usually use statistical methods to track. On combined trackers, methods of integrating above-mentioned designs are discussed. Though tracking methods possess respective merits, drawbacks are accompanied with them. It is easy to see that, recent development walks towards methods using higher dimensions to describe objects.

1.2.4 Motion Estimation

Some tracking methods use motion models to make searching efficient. Thanks to Newtonian mechanics, object motion is predictable. Nevertheless, in practical situations, noises damage object motion estimation. Once observation made in previous frame was mixed with noises, estimation is then deviated. Finally, error accumulation breaks the system. Correct motion estimation helps object tracking in computation reduction, but it also increases the risk of false positive.

Several object tracking approaches have been addressed from object representations to tracking methods. Different approaches are suitable for different applications. In next section, developments of tracking methods are discussed.

1.3 Developments of Tracking Methods

Yang [11] has mentioned that conventional tracking methods use prediction and verification. At previous frame, frame $t-1$, tracker makes predictions for next state. Sampling and particle filtering are available prediction methods. At frame t, verification of previous predictions is

realized by observations. The overall architecture is just like recursive process of Kalman filter, prediction and correction [12]. Most of these methods require offline training and do not have high-level notion of objects. Also, they possess the same appearance model throughout frame sequences.

Recently, tracking algorithms are trying to break the limitation of using constant appearance model. Continuously learning and updating appearance model have made trackers endure large illumination and pose variation. In this way, tracking methods become more robust to environmental variation. Yang [11] has pointed out that several tracking algorithms have applied this concept, including generative or discriminative algorithm, multiple instance learning and articulating object tracking. In this thesis, we describe some of them in related works.

1.4 Contribution

In this paper, we propose an enhanced tracking method called EOBT, which is based on Online Boosting for Tracking (OBT) [13][14]. OBT is an amazing tracking method which has ability to adapt those variations between frames. It is also a model-less tracking methods without motion estimation. However, the most crucial problem is also due to its excellent adaptability, which is called the template update problem, or drifting. This is a stability-plasticity dilemma and has been addressed in [15]. In short, increasing adaptive might causes stability drop. We import depth information to enhance OBT on drifting-resistance ability. Also, we introduce scalability for EOBT, hoping to reduce background noises caused by distance variations. To solve temporary occlusion problem, we design a new mechanism called lifetimer for our tracker. With lifetimer, tracker is able to stop updating when getting lost. Meanwhile, it differentiates temporary occlusion from disappearance. This is important for practical usage to notify system when should stop tracking.

1.5 Synopsis

The remaining part of this paper is organized as follows. Chapter 2 introduces OBT in details. Related works designed to solve drifting problem are described in chapter 3, including online semi-supervised boosting for tracking [16], beyond semi-supervised tracking [17] and tracking with online multiple instance learning [18]. In chapter 4 and 5, we propose our enhanced tracking method and demonstrate some experiments. Chapter 6 discusses and compares our methods with those in related works. Finally, conclusion summarizes our research in chapter 7. More information about boosting, online boosting, semi-supervised learning and multiple instance learning are attached in appendices.

Chapter 2

Background

In this chapter, the tracking method, OBT, is described in detail. It has been used in related works and proposed method. The reason why the online boosting algorithm is adopted in proposed method is mentioned first. Then, the online boosting algorithm used for feature selection and object tracking comes after. For more information about the boosting and online boosting algorithm, please refer to appendices.

2.1 Preliminary

Boosting is an ensemble learning algorithm used to distinguish one category from another. It has been researched theoretically and experimentally for past two decades [19] [20]. Research shows that, boosting can be interpreted as additive logistic regression [20], which means that the loss function of boosting is an exponential function and boosting can be regarded as a greedy learning method. Also, in the boosting algorithm, margins between categories keep increasing during consecutive iterations even when classification is finished. These two merits have made boosting widely applied into diverse research areas. For example, Viola and Jones [21] apply boosting algorithm to face detection and achieve remarkable success.

Originally, the boosting algorithm ,also known as batch boosting, uses all training samples in one iteration. To increasing its applicability, the batch boosting algorithm has been further developed in an online manner, called online boosting. In the online boosting algorithm, a training sample is only used once and discarded forever. In this manner, online boosting algorithm

is appropriate for real-time applications. However, the major challenge is that, the hypotheses returned from online boosting algorithm may not be identical to those returned from batch boosting algorithm.

To solve this inconsistency, Oza [22] modifies the weight adjustment scheme in batch boosting algorithm and proves that using *lossless* learning algorithm as base model, the hypotheses returned from online boosting would converge to those returned from batch boosting. Lossless learning algorithms are described in appendices. Concerning about theories of convergence, please refer to Oza's PhD thesis [22]. With Oza's achievement, Grabner and Bischof [13] leverage the online boosting algorithm for feature selection and object tracking, as described in 2.2 and 2.3.

2.2 Online Boosting for Feature Selection

In 2006, Grabner and Bischof have pointed out that, in the online boosting algorithm, the importance adjustment of samples is modified. Single sample is propagated through all base models. This modification solved the crucial problem of unknown weight distributions of entire training samples. Therefore, Grabner and Bischof applied online boosting algorithm and proposed a novel feature selection method.

Grabner et al. [14] [13] have introduced *selectors* into their algorithm. The selector selects the best weak classifier from global weak classifier pool [14]. In [13], several local weak classifier pools are used for determining a selector, while the local weak classifier pools are replaced with one global weak classifier pool in [14] for better performance. Please refer to the illustration shown in Figure 2.2. We consider that, the mechanism provided by selectors is similar to cascade structure designed by Viola and Jones [21]. See Figure 2.1.

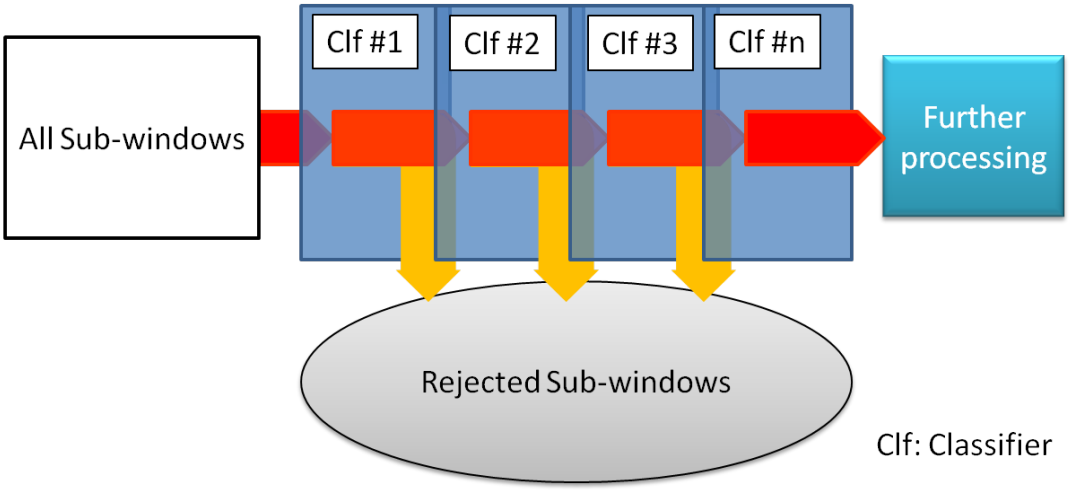


Figure 2.1: Scheme of detection cascade. Every sub-window is detected by every classifier (Clf). If it does not conform detection rules of classifiers, it is rejected right away and never be used again.

The importance of sample (λ) is initialized to 1. When one sample enters, all weak classifiers are used to judge this sample. After judgement, errors of weak classifiers (e_1, e_2 ... e_n) are evaluated. The weak classifier with the lowest error is selected as the first selector. Then, the voting weight (α_i, where i is the index of the selector) of the selector is calculated according to its error and can be represented as

$$\alpha_i = \frac{1}{2} \times \ln(\frac{1 - e_i}{e_i}). \tag{2.1}$$

Next, the importance of the sample is adjusted for the next selector (see the following equations) according to the error e_{i-1} of the previous selector.

if the judgement of previous selector is correct,

$$\lambda = \lambda \times \frac{1}{2 \times (1 - e_i)} \tag{2.2}$$

else

$$\lambda = \lambda \times \frac{1}{2 \times (e_i)} \tag{2.3}$$

end if

After calculating its voting weight and adjusting importance, the sample with the adjusted importance is propagated to the next selector. When completing selecting weak classifiers for

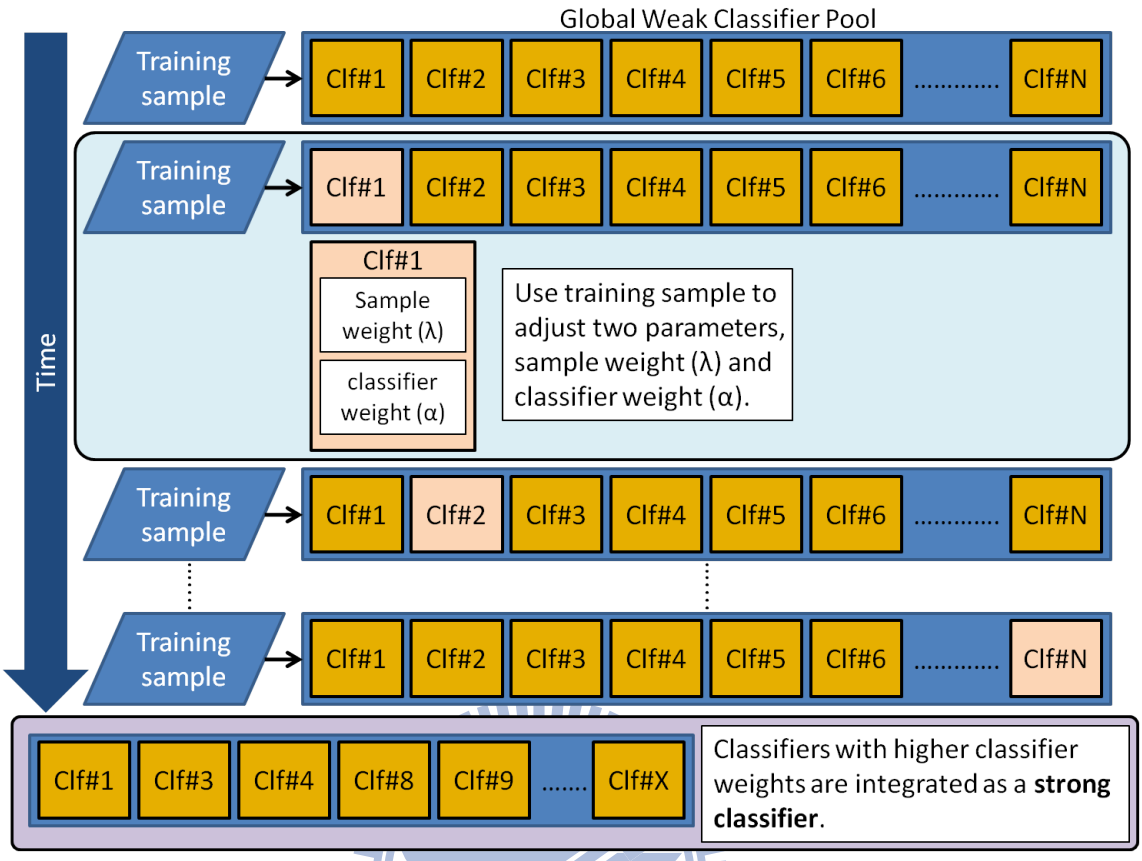


Figure 2.2: The online boosting algorithm for feature selection. Every classifier in global weak classifier pool are trained using training samples. The sample weight and classifier weights are adjusted during training. Eventually, those classifiers with higher classifier weights are selected and integrated as a strong classifier.

all selectors, these selectors (h_i^{sel}) are then merged to form a strong classifier (h^{strong}).

$$h^{strong} = sign(\sum_{i=1}^{N} \alpha_i \times h_i^{sel}(x)), \tag{2.4}$$

where α_i is the voting weight of the i th selector, and $sign$ function is defined as

$$sign(x) = \begin{cases} -1, & \text{if } x < 0, \\ 0, & \text{if } x = 0, \\ 1, & \text{if } x > 0. \end{cases} \tag{2.5}$$

2.3 Online Boosting for Tracking

To make use of online boosting for object tracking, Grabner et al. have designed a new procedure. There are two stages in this procedure, training stage and tracking stage. We illustrate them separately as follows.

2.3.1 Training Stage

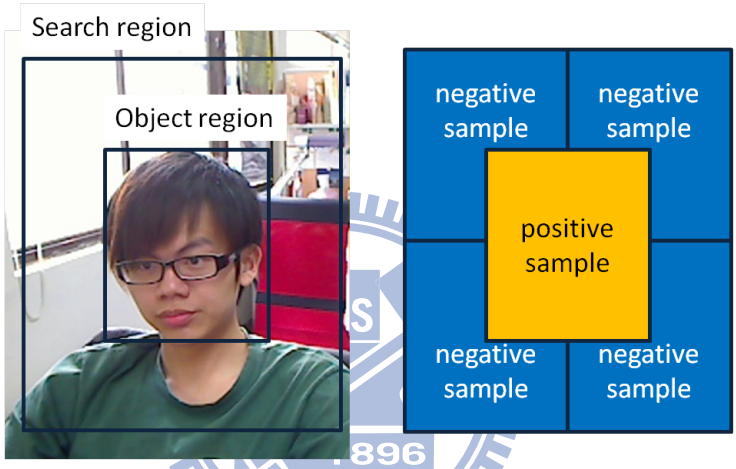


Figure 2.3: Tracking by classification. The object region is selected as a positive sample. Neighbouring regions in search region is selected as negative samples.

The goal for training stage is to build a tracker for tracking. To answer the needs of object tracking, Grabner et al. [13] use the tracked object as a positive sample and surrounding background as negative samples. See Figure 2.3. They only use first frame for training because the ROI is decided by user and can be fully trusted. Since online boosting is a kind of supervised learning, the first frame is regarded as his teacher for tracking on next frame. After using online boosting for feature selection, a strong classifier distinguishes tracked object from background is made. This strong classifier could be regarded as a tracker for further usage.

2.3.2 Tracking Stage

After constructing the strong classifier for tracked object, first frame is then discarded. On next frame, search region is specified according to the previous tracked object position. See Figure 2.4 for better illustration. The tracker created by previous frame is used scanning every position in the search region. While scanning, the tracker evaluates every position and produce hypotheses. These hypotheses estimate in what degree of confidence that the corresponding position is the tracked object. In other words, hypothesis is a kind of similarity estimation. A confidence map is created after scanning, see right image on second row of Figure 2.4. The position with highest confidence is chosen as new object position. This procedure is iteratively

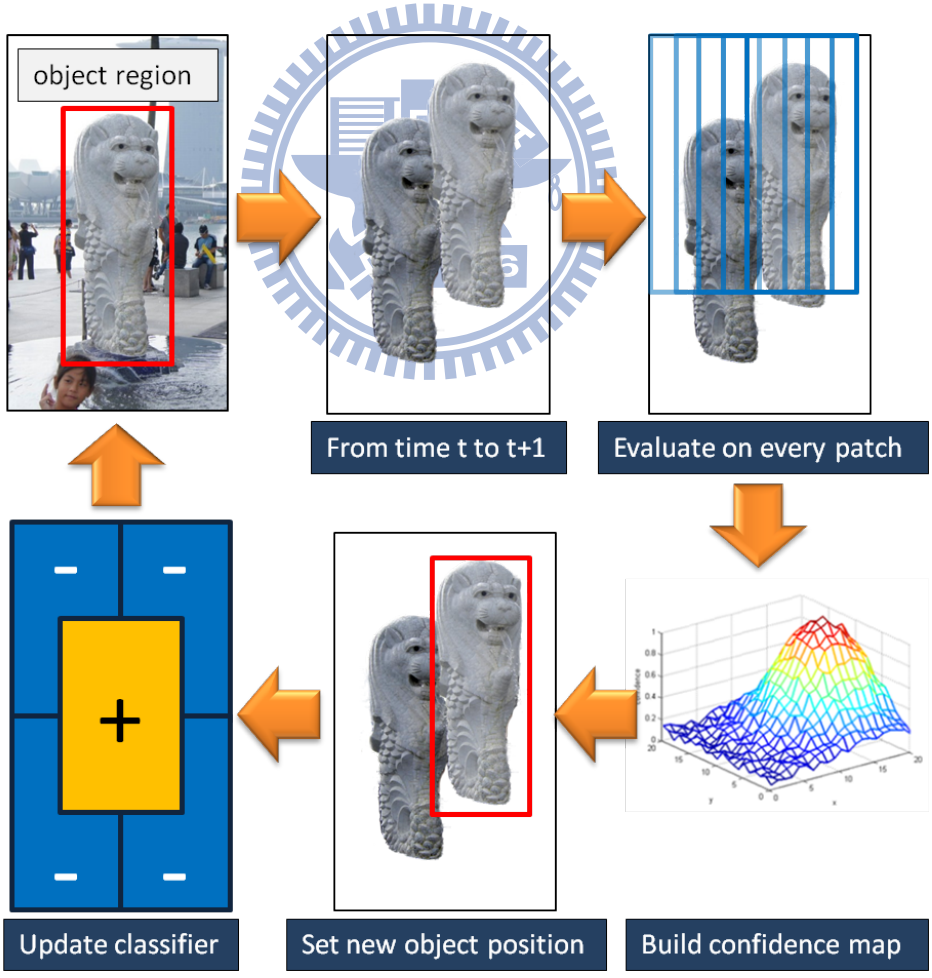


Figure 2.4: Algorithm of online boosting for tracking. The classifier is updated every time the picture is captured and the update method is based on confidence map.

(a) frame #199 (b) frame #200 (c) frame #201 (d) frame #202

Figure 2.5: Example of drifting problem. From (a) to (d) are consecutive frames. From (b) to (c), the tracker drifts because the tracked lady slightly moves away from camera.

applied to every consecutive frame. Hence, object tracking could be achieved.

Before tracking on next frame, the tracker should be updated according to new object position. This step makes tracker has adaptability to adapt variations. Another thought is that, in training stage, a tracker is made by constructing an internal appearance model. The tracker uses this internal appearance model to match with new coming frames. Since online boosting for tracking is a supervised learning process, the tracker should trust new object position determined by itself. New object position is regarded as a new teacher and used to update tracker's internal appearance model. This step is also the reason why drifting problem occurred. Because position estimation is not always correct, slight errors come in. These slight errors are learnt by tracker. Hence, errors are accumulated during object tracking, causing tracker lost or to be mistrusted. See Figure 2.5.

For better illustration of online boosting for tracking, we put a system overview in Figure. 2.6. Two stages form the system, training stage and tracking stage. In training stage, first gray image is used to generate a tracker. This tracker is then used to track on the first image several times for training. Number of training iteration can be set by user. In tracking stage, the trained tracker is used on succeeding frame sequence. Similar process is carried out on all frame sequence. If confidence of tracker is under 0, that means the tracker is lost. The tracking process exit if tracker lost or all frames are been tracked.

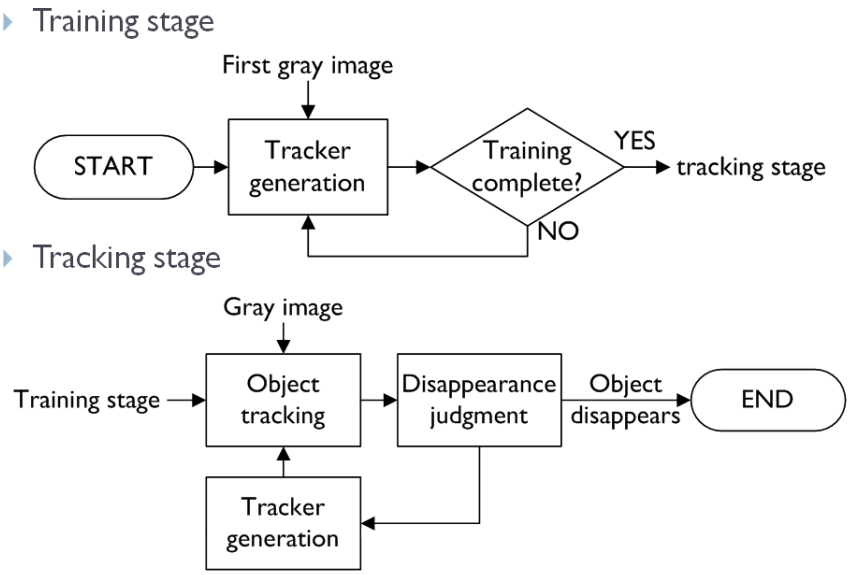


Figure 2.6: System overview of online boosting for tracking.

2.4 Summary

In summary, online boosting for tracking is a two-stage tracking method. First, internal template is constructed in training stage. Online boosting for feature selection is used to train the tracker. Next, in tracking stage, the tracker is used to search new object position by selecting the position with best confidence. Then tracker fully trusts this new object position and update internal appearance model according to it. This self learning is classified as supervised learning strategy. One of the key problems of this strategy is error accumulation. In object tracking applications, error accumulation causes drifting. In next chapter, several researches on drifting are stated in company with pros and cons. Different learning strategies are also been used to suppress drifting.

Chapter 3

Related Work

In this chapter, three drifting-suppressing methods are introduced. Our scope is limited in model-less tracking methods without motion estimation, and including appearance updating skills. Specifically, we adopt discriminative tracking methods, not generative ones. For generative tracking methods, please refer to [23] or [24]. Here in this chapter, we introduce online semi-supervised boosting for tracking [16], beyond semi-supervised tracking [17], and tracking with online multiple instance learning [18] as follows.

3.1 Online Semi-Supervised Boosting for Tracking (OSSB)

Semi-supervised learning is a learning strategy exploiting not only known data, but also unknown data for learning. Recently semi-supervised learning has gained a lot of attention because known data collection is always a tedious work. For tracking applications like human tracking for example, labelling people in every frame is an inevitable preprocessing work. To exploit known data, researchers have proposed many methods to bridge the gap between known and unknown data. Xiaojin has maintained a comprehensive survey about semi-supervised learning on his website [1] [25]. Since Mallapragada's version [26] of semi-supervised learning has been adopted by Grabner [16], we illustrate Mallapragada's theory in appendices. Here we only introduce how Grabner et al. [16] make use of their semi-supervised learning strategy on object tracking.

The key problem Grabner et al. solved is drifting problem. Grabner believes that, for-

[1] http://pages.cs.wisc.edu/ jerryzhu/research/ssl/semireview.html

mulating the update process in a semi-supervised fashion could significantly alleviate drifting problem [16]. They combine decision of a given prior classifier, $H^P(x)$, and an online classifier, $H_n(x)$, to estimate object position. On the first frame, the prior classifier is trained using method of original OBT. This prior classifier can be regarded as the teacher in all tracking process. Other frames are belongs to unknown data, which needs to be classified using semi-supervised learning. Formally, the Mallapragada's deduction on SemiBoost told us that the best weak classifier could be obtained using

$$h_n = \arg\min_{h_n} \left(\frac{1}{|\chi^L|} \sum_{\substack{x \in \chi^L \\ h_n(x) \neq y}} \omega_n(x, y) - \frac{1}{|\chi^U|} \sum_{x \in \chi^U} (p_n(x) - q_n(x)) \alpha_n h_n(x) \right), \tag{3.1}$$

where

$$p_n(x) = e^{-2H_{n-1}(x)} \frac{1}{|\chi^L|} \sum_{x_i \in \chi^+} S(x, x_i) + \frac{1}{|\chi^U|} \sum_{x_i \in \chi^U} S(x, x_i) e^{H_{n-1}(x_i) - H_{n-1}(x)}, \tag{3.2}$$

$$q_n(x) = e^{2H_{n-1}(x)} \frac{1}{|\chi^L|} \sum_{x_i \in \chi^-} S(x, x_i) + \frac{1}{|\chi^U|} \sum_{x_i \in \chi^U} S(x, x_i) e^{H_{n-1}(x) - H_{n-1}(x_i)}, \tag{3.3}$$

and

$$\omega_n(x, y) = e^{-2yH_{n-1}(x)}. \tag{3.4}$$

The weight α_n can be obtained by taking derivative of Eq.3.1 with respect to α_n and setting it to zero [27], where

$$\alpha_n = \frac{1}{4} \ln \left(\frac{\frac{1}{|\chi^U|} \left(\sum_{\substack{x \in \chi^U \\ h_n(x)=1}} p_n(x) + \sum_{\substack{x \in \chi^U \\ h_n(x)=-1}} q_n(x) \right) + \frac{1}{|\chi^L|} \sum_{\substack{x \in \chi^L \\ h_n(x)=y}} \omega_n(x, y)}{\frac{1}{|\chi^U|} \left(\sum_{\substack{x \in \chi^U \\ h_n(x)=1}} q_n(x) + \sum_{\substack{x \in \chi^U \\ h_n(x)=-1}} p_n(x) \right) + \frac{1}{|\chi^L|} \sum_{\substack{x \in \chi^L \\ h_n(x) \neq y}} \omega_n(x, y)} \right). \tag{3.5}$$

Especially, similarity could be made by a boosting classifier, where

$$S(x_i, y_j) \approx H^{sim}(x_i, y_j). \tag{3.6}$$

For positive samples, $\sum_{x_i \in \chi^+} H^{sim}(x, x_i)$ is a probability measure that x corresponds to the positive class. We could express as

$$\sum_{x_i \in \chi^+} H^{sim}(x, x_i) \approx H^+(x), \tag{3.7}$$

also,

$$\sum_{x_i \in \chi^-} H^{sim}(x, x_i) \approx H^-(x). \tag{3.8}$$

Here, Grabner et al. use the prior classifier to measure similarity, so the probability measure of positive and negative samples is directly replaced by $H^P(x)$, i.e., $H^+(x) \sim H^P(x)$ and $H^-(x) \sim 1 - H^P(x)$. On the other hand, since boosting could be viewed as additive logistic regression by stage wise minimization of the exponential loss $L = \sum_{x \in \chi^L} e^{-yH(x)}$ and confidence measure is

$$P(y = 1|x) = \frac{e^{H(x)}}{e^{H(x)} + e^{-H(x)}}. \tag{3.9}$$

Therefore, Eq.3.2 and Eq.3.3 are simplified as

$$\tilde{p_n}(x) \approx e^{-H_{n-1}(x)} \sum_{x_i \in \chi^+} S(x, x_i) \approx e^{-H_{n-1}(x)} H^+(x) \approx \frac{e^{-H_{n-1}(x)} e^{H^P(x)}}{e^{H^P(x)} + e^{-H^P(x)}} \tag{3.10}$$

and

$$\tilde{q_n}(x) \approx e^{H_{n-1}(x)} \sum_{x_i \in \chi^-} S(x, x_i) \approx e^{H_{n-1}(x)} H^-(x) \approx \frac{e^{H_{n-1}(x)} e^{-H^P(x)}}{e^{H^P(x)} + e^{-H^P(x)}} \tag{3.11}$$

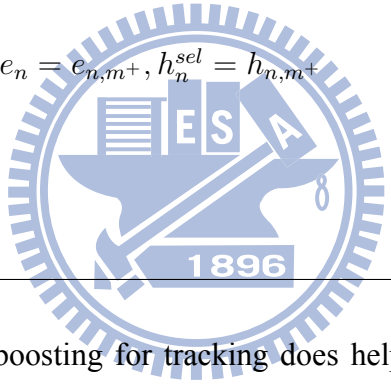
Because discriminative classifier is used, the interest would be put on their difference, $\tilde{z_n}(x)$, which Grabner et al. named as "pseudo-soft-label."

$$\tilde{z_n}(x) = \tilde{p_n}(x) - \tilde{q_n}(x) = \frac{\sinh(H^P(x) - H_{n-1})}{\cosh(H^P(x))} = \tanh(H^P(x)) - \tanh(H_{n-1}(x)). \tag{3.12}$$

We put the algorithm of online semi-supervised boosting for feature selection in Algorithm 1. Also, for better illustration, we adopt the SemiBoost concept from [27] in Figure.3.1.

Algorithm 1 Algorithm 1. On-line Semi-supervised Boosting for feature selection

Require: training (labeled or unlabeled) example <x,y>, $x \in \chi$
Require: prior classifier H^P (can be initialized by training on χ^L)
Require: strong classifier H (initialized randomly)
Require: weights $\lambda^c_{n,m}$, $\lambda^\omega_{n,m}$ (initialized with 1)
  for $n = 1, 2, ..., N$ **do**
    if $x \in \chi^L$ **then**
      $y_n = y, \lambda_n = exp(-yH_{n-1}(x))$
    else
      $y_n = sign(p(x) - q(x)), \lambda_n = |p(x) - q(x)|$
    end if
    for m=1,2,...,M **do**
      $h_{n,m} = update(h_{n,m}, < x, y >, \lambda)$
      if $h^{weak}_{n,m}(x) = y$ **then**
        $\lambda^c_{n,m} = \lambda^c_{n,m} + \lambda_n$
      else
        $\lambda^\omega_{n,m} = \lambda^\omega_{n,m} + \lambda_n$
      end if
      $e_{n,m} = \frac{\lambda^\omega_{n,m}}{\lambda^c_{n,m}+\lambda^\omega_{n,m}}$
    end for
    $m^+ = argmin_m(e_{n,m}), e_n = e_{n,m^+}, h^{sel}_n = h_{n,m^+}$
    if $e_n = 0 or e_n > \frac{1}{2}$ **then**
      exit
    end if
    $\alpha_n = \frac{1}{2} \times \ln\{\frac{1-e_n}{e_n}\}$
  end for



Using semi-supervised boosting for tracking does help alleviate drifting. However, the problem is its applicability seems also been limited. The reason is that, the prior classifier is fixed throughout the frame sequence. Each time when position estimation is needed, the prior classifier is recalled to mutually decide position. However, if a rotating object is tracked, this limited applicability breaks the tracking system. Tracker gets lost once the tracked object starts to rotate. Therefore, semi-supervised tracker gets lost in this situation. We may conclude that, when appearance of tracked object is always the same, online semi-supervised tracking method is robust. While rotating object is tracked, this tracking method might be no longer applicable.

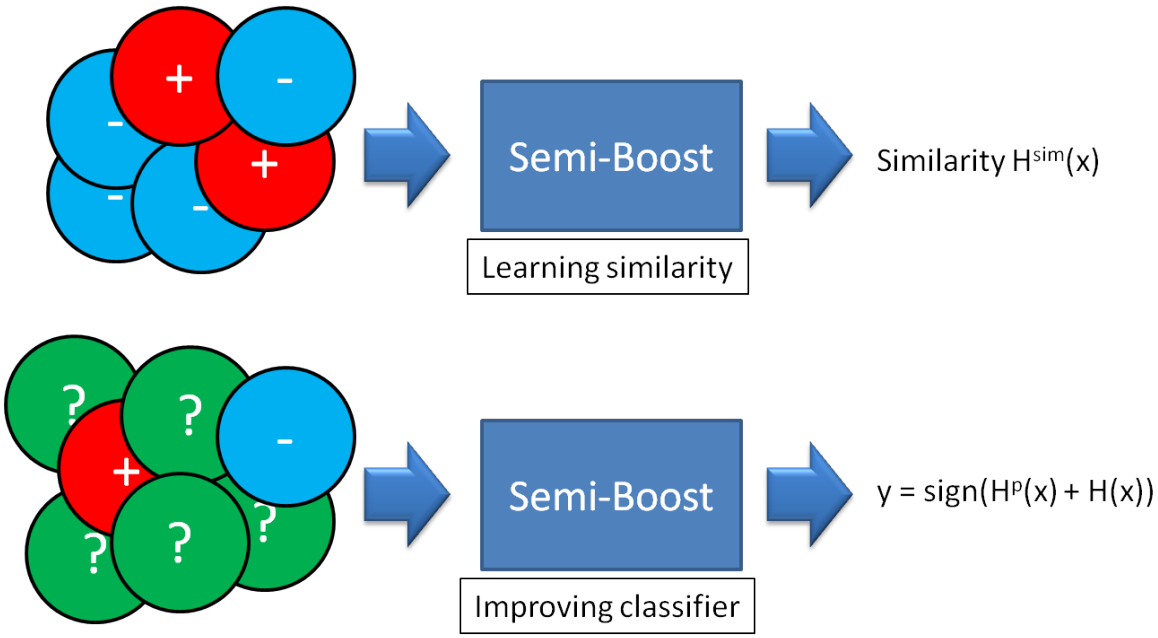


Figure 3.1: SemiBoost used for object tracking. Originally, semi-boost is used for machine learning via calculating similarity. The author makes use of similarity on object tracking.

3.2 Beyond Semi-Supervised Tracking (BSST)

Beyond semi-supervised tracking is proposed by Stalder et al. in 2009 [17]. Basically, they focused on extending semi-supervised learning. They pointed out that OSSB has two drawbacks. We list them as follows.

- Influence of prior classifier might not be optimal, especially in the case of partial occlusion.
- The prior classifier does not specialize to a specific object, i.e., it cannot recognize similar objects.

These two problems have been properly solved by Stalder's architecture. Please refer to Figure 3.2.

In Figure 3.2, (a) is a pure detection process. A filled circle is represented as one detection. Since the appearance model used to detect is unchanged, no drifting occurred. In (b), conven-

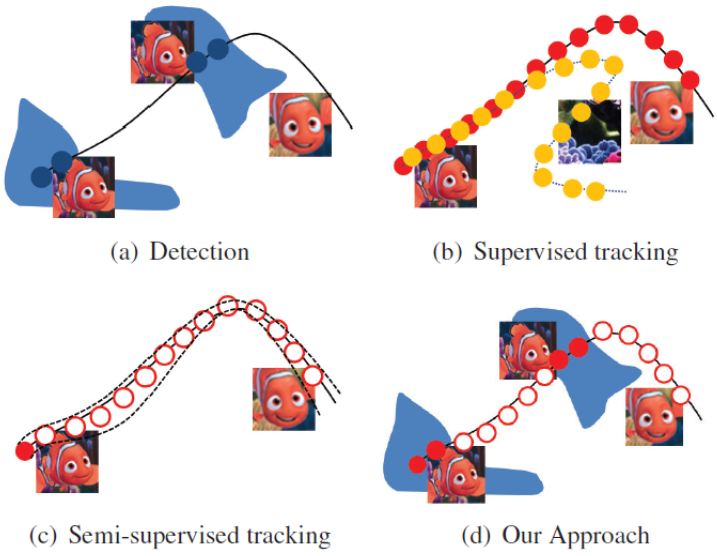


Figure 3.2: Different adaptation strategies. Adopted from [17].

tional supervised learning strategy is applied. The same route as its counterpart in (a) is ideal object track. However, supervised tracking is suffered from drifting problem. In (b), the tortuous route represents drifting situation. In (c), semi-supervised learning strategy corrects drifting by fixed prior classifier. The filled circle is regarded as prior classifier training, and the other unfilled circles are viewed as semi-supervised learning. Nevertheless, limited adaptability of fixed prior classifier hurts its plasticity. In (d), Stalder et al. have made prior classifier adaptive conservatively. Those filled circles are tracking with updating both prior classifier and online classifier. Unfilled circles are tracking with only updating online classifier.

The overall architecture could be divided into three parts, detector, recognizer and tracker, as Figure 3.3 shown. All of them are classifiers with different level of adaptability. The detector is offline trained classifier and without updating while tracking. The recognizer is a supervised online classifier with conservative updating. Here we briefly introduce overall mechanism of beyond semi-supervised tracking. During initialization, detector, recognizer and tracker are trained using first frame. During object tracking, semi-supervised tracking process is applied

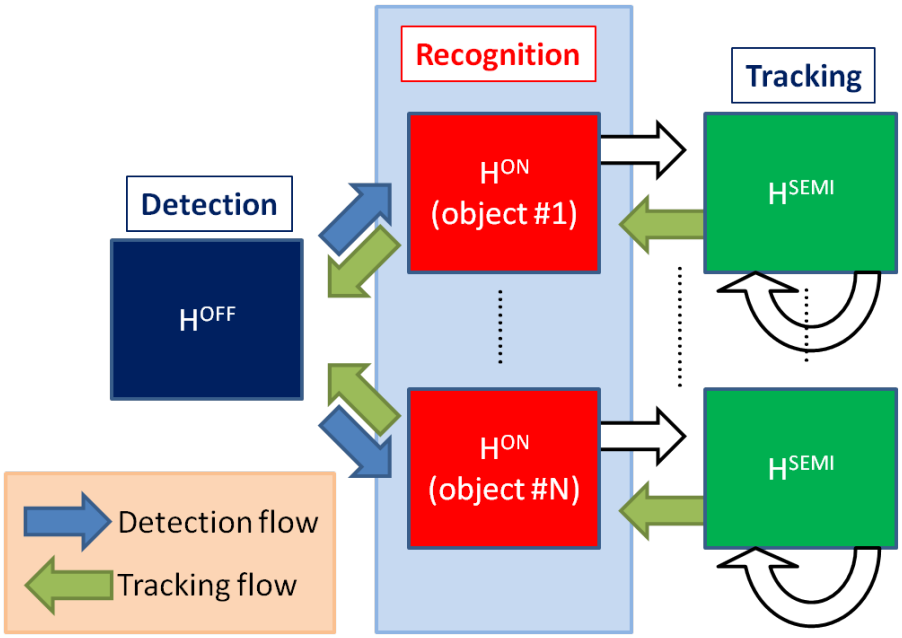


Figure 3.3: Architecture of beyond semi-supervised tracking.

with recognizer and tracker, which recognizer is acted as prior classifier and tracker is as online classifier. If tracking is successful, prior and online classifier are not updated directly. They use detector to examine the same position again to verify if it is an appropriate appearance. If so, prior and online classifiers are updated. If not, only online classifier is updated. On the other hand, if tracking failed, prior and online classifier are recreated. Then, detector is used to find a potential position on the next frame. If detector found, the prior and online classifiers are reinitialized again. If not, this frame is discarded and keeps detecting until tracked object is found. We have summarized their tracking process in Figure 3.4. Please refer.

Because detector guarantees a fixed false positive and detection rate, drifting could be detected during tracking. We consider that, detector is as a supervisor, which suppresses drifting probability. However, beyond semi-supervised tracking has the same limitation as semi-supervised tracking. Though beyond semi-supervised tracking does extend semi-supervised tracking, the overall bottleneck is locked by detector. If arbitrary object tracking is asked, constructing specific detector is still a tedious work. This tracking method is applicable for conven-

tional object tracking, like human tracking.

3.3 Online Multiple Instance Learning (OMIL)

Multiple instance learning (MIL) has been studied for decades. MIL learns a concept from multiple instances, whereas conventional object tracking methods directly learn from instances. MIL learning strategy is fairly reasonable because it contains a certain degree of ambiguities. For example, see Figure 3.5. These labellings are part of correct tracked object, but with different level of label noises. The ambiguity causes target to drift.

The most substantial contribution is that MIL offers bag probabilities. Here, *bag* could be regarded as *concept* described before. Bag is a set of instances and defined according to its labelling as follows.

> A bag is labelled positive even if only one of the instances in it falls within the concept. A bag is labelled negative only if all the instances in it are negative [28].

We describe theories of multiple instance learning in appendices. Here, we briefly illustrate how

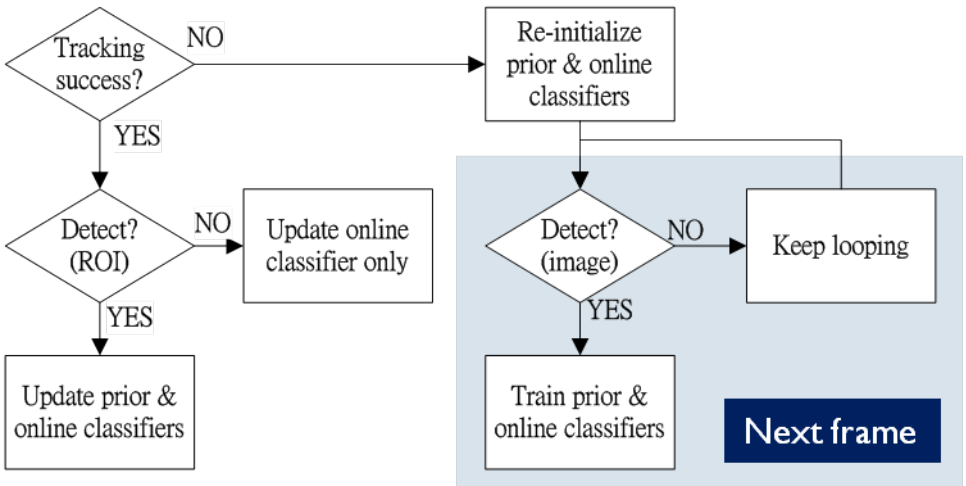


Figure 3.4: Tracking process of beyond semi-supervised tracking.

Babenko et al. modify MIL so as to apply it on object tracking.

Babenko et al. modify MIL to online MIL (OMIL) for object tracking, which combines online boosting [22] and MIL [29]. For online boosting, since we have described some of them in Background and introduced them in detail in appendices, we solely focus on revisions they made. The authors choose MILBoost [29] as their MIL learning strategy. Deducted from MILBoost, since boosting can be viewed as an additive logistic regression, the log likelihood of bag of MILBoost is,

$$\log \mathcal{L} = \sum_i \Big(\log p(y_i|X_i) \Big). \tag{3.13}$$

The bag probability model they used is Noisy-OR model, which adopted from [29], is as follows.

$$p(y_i|X_i) = 1 - \prod_j \Big(1 - p(y_i|x_{ij}) \Big) \tag{3.14}$$

where $p(y_i|X_i)$ is a bag probability, and $p(y_i|x_{ij})$ is probability of instances.

Because the logistic regression is executed by weak classifiers, they choose weak classifiers according to loss function $\mathcal{J}$, i.e.,

$$(\mathbf{h}_k, \alpha_k) = \arg\max_{\mathbf{h}\in\mathcal{H},\alpha} \mathcal{J}(\mathbf{H}_{k-1} + \alpha\mathbf{h}), \tag{3.15}$$

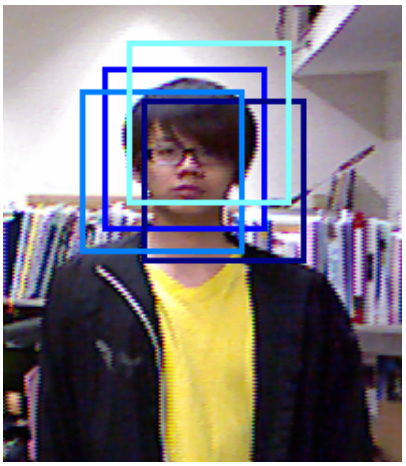


Figure 3.5: Different labellings (instances) for the same target.

Algorithm 2 Algorithm 2. Online Multiple Instance Learning (OMIL)

Input: Dataset $\{X_i, y_i\}_{i=1}^{N}, where X_i = \{x_{i1}, x_{i2}, ...\}, y_i \in \{0, 1\}$
Update all M weak classifiers in the pool with data $\{x_{ij}, y_i\}$.
Initialize $H_{ij} = 0$ for all i, j
for $k = 1$ to K **do**
  for $m = 1$ to M **do**
    $p_{ij}^m = \sigma\big(H_{ij} + h_m(x_{ij})\big)$
    $p_i^m = 1 - \Pi_j(1 - p_{ij}^m)$
    $\mathcal{L}^m = \Sigma_i\big(y_i \log(p_i^m) + (1 - y_i) \log(1 - p_i^m)\big)$
  end for
  $m^* = argmax_m \mathcal{L}^m$
  $\mathbf{h}_k(x) \leftarrow h_{m^*}(x)$
  $H_{ij} = H_{ij} + \mathbf{h_k(x)}$
  Output: Classifier $\mathbf{H}(x) = \Sigma_k \mathbf{h}_k(x), where p(y|x) = \sigma(\mathbf{H}(x))$
end for

where $\mathbf{H}_{k-1}$ is a strong classifier made by previous (k-1) weak classifiers, and $\mathcal{H}$ is all possible weak classifiers. They model instance probability using sigmoid function as

$$p(y|x) = \sigma\big(\mathbf{h}(x)\big), \tag{3.16}$$

where $\sigma(x)$ is a sigmoid function. The bag probability is directly adopted from Eq. 3.14 Here they have slightly modified Eq. 3.15, which absorbed scalar weight α. The overall online MIL-Boost algorithm is depicted in Algorithm 2.

OMIL algorithm also uses architecture of online boosting for feature selection. The inner for-loop evaluates on entire global weak classifier pool, which contains M weak classifiers. The p_{ij}^m and p_i^m correspond to instance probability and bag probability. After evaluation on all weak classifiers, $\mathbf{h}_k(x)$ picks the weak classifier with maximum loss, see Eq. 3.13. Here, $\mathbf{h}_k(x)$ could be viewed as a selector. The selector is combined into stage-wise strong classifier H_{ij}. After deciding all k selectors, the strong classifier $\mathbf{H}(x)$ is a linear combination of k selectors and bag probability $p(y|x)$ is directly obtained using sigmoid function $\sigma(x)$.

Babenko et al. use Haar-like feature with four parameters $(\mu_1, \sigma_1, \mu_0, \sigma_0)$. The weak clas-

sifier is defined as

$$h_k(x) = \log\left[\frac{p_t(y=1|f_k(x))}{p_t(y=0|f_k(x))}\right], \tag{3.17}$$

where $p_t(f_t(x)|y=1) \sim \mathcal{N}(\mu_1, \sigma_1)$. Their update rules are

$$\mu_1 \leftarrow \gamma\mu_1 + (1-\gamma)\frac{1}{n}\sum_{i|y_i=1} f_k(x_i) \tag{3.18}$$

$$\sigma_1 \leftarrow \gamma\sigma_1 + (1-\gamma)\sqrt{\frac{1}{n}\sum_{i|y_i=1}(f_k(x_i)-\mu_1)^2} \tag{3.19}$$

Similar manner is applied on $y=0$.

OMIL is an excellent object tracking in that, it interprets object from the notion of *object*. This means that it has excellent performance on temporary partial occlusion and maintain at good object position. This tracking method is especially applicable for human tracking while appearance changes, like wearing a hat, is normally happened. Another merit is that, this tracking method seems have the ability returning to correct position. Please refer to their website [2] seeing the experimental video 'David.' Their tracker is not correctly aligned with David's face at frame #300. However, at frame #406 when David wears back his glasses, the tracker is aligned correctly again. On the other side, drifting can still be found in experimental videos, like 'Occluded Face.' We consider that OMIL has turned its attention to the book at frame #670 to #720. The same stability-plasticity problem has also struck at OMIL. Especially, in 'Coke Can', OMIL has already out of focus and turned its attention to operator's hand. This problem cannot be found from the position error on error plot in [18].

In next chapter, we proposed our own tracking method with depth info. We also proposed our evaluation method to repair shortcomings of position error.

[2] http://vision.ucsd.edu/ bbabenko/project_miltrack.shtml

Chapter 4

Enhanced OBT

In this paper, we propose a novel method for eliminating drifting problem. Since all related works are developed from OBT, and OBT has the ability to get through manifold problem, we choose OBT as our tracking method and enhance it. We illustrate proposed method in this chapter and discuss manifold consideration in discussion. Our approach, EOBT, consists of three mechanisms, which introduces depth, multiple scales and lifetimer to OBT. We describe them separately in the following sections.

4.1 Depth

We enhance original OBT with depth info. Thanks to Microsoft®XBOX 360, a compact solution on both color and depth info can be caught simultaneously using Kinect. The original OBT does not include depth information. We introduce depth to distinguish object from background, hoping to eliminate background interference.

In addition, the precision of depth info returned from Kinect is 16 powers of 2. We have normalized to 0-255 for convenience, not only for human reading but also stable for machine processing. We called this normalized depth values as a depth image. For succeeding paragraph, we use the word, *depth images*, to represent depth info.

Our goal is to exploit depth info. We consider that direct acquiring image within specific depth range helps reducing the complexity of tracking environment. Hence, according to the depth of tracked object, we can take images out with the same depth and track only on these

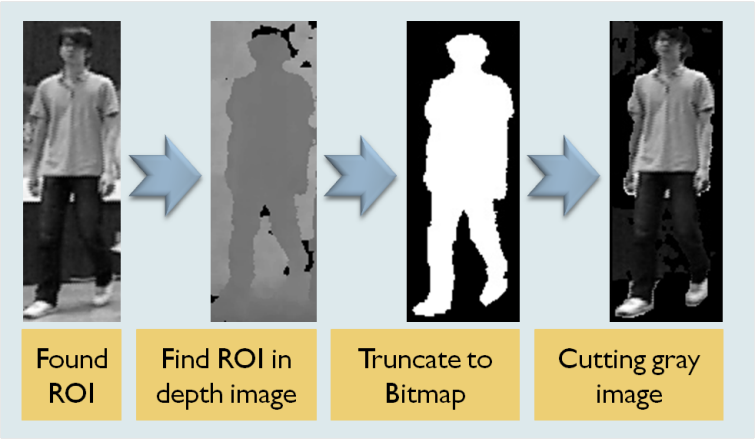


Figure 4.1: Truncation process of EOBT.

segmented images. This filtering process is illustrated in Figure 4.1.

In the beginning, tracked object, the ROI, is set by user. Next, find the same ROI on depth image. Use truncation methods to isolated tracked object from background and transform this concept into a filter. Finally, use this filter to filter out tracked object. Here, truncation method is a critical problem. Figure 4.2 illustrates this problem. In Figure4.2, the top plot is depth

Figure 4.2: Truncation problem.

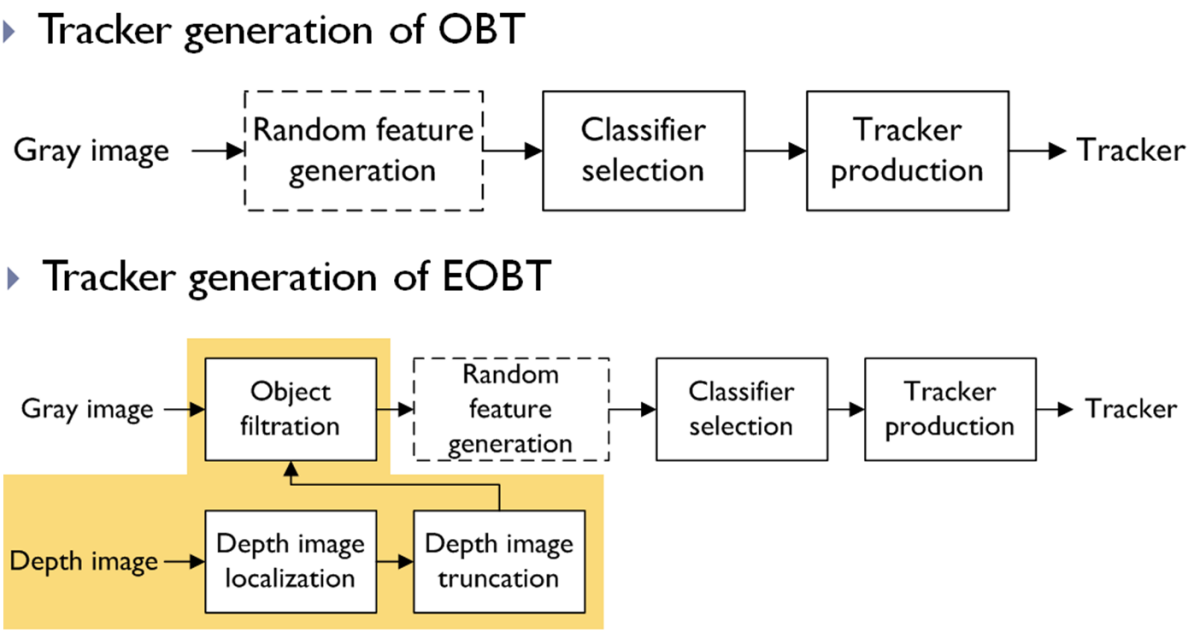


Figure 4.3: Tracker generation of OBT and EOBT.

histogram of ROI. Depth range between 110 and 129 is desired truncation range. Middle image shows truncation result. Below 110 or over 129 gives bad truncation results, as shown in left and right images. There are several different truncation methods, such as fixed boundary, average value, statistics and value at middle point, etcetera. Since truncation methods could be taken as an optional choice, we let user selects his/her favour.

To show which part we revised, Figure 4.3 compares tracker generation of EOBT with OBT. EOBT has inserted object filtration to gray image. The object filtration uses the filter produced by binarizing depth image. The other parts of tracker generation remain the same. This slight revision promises that processing speed is not diminished too much.

Figure 4.4: The need for scalability.

4.2 Multiple scales

If tracker size remains unchanged, it increases probability of drifting because of too much background interferences in tracker. This problem is hard for OBT since there is no clue for distinguishing appearance changes from object distance change. See Figure 4.4. Since EOBT has exploited depth info and isolated tracked object from background, we add scalability for EOBT.

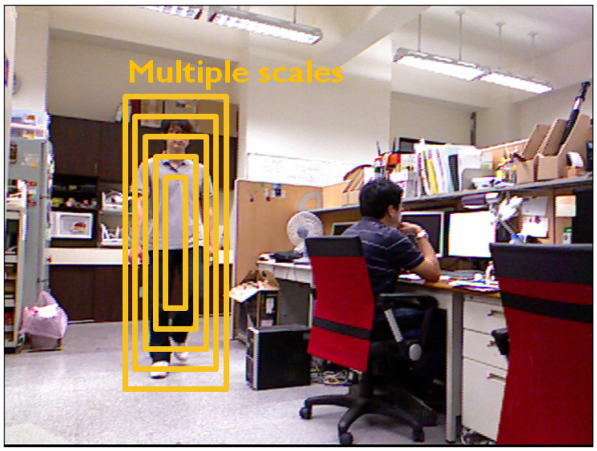


Figure 4.5: Scalable adjustment using multiple scales.

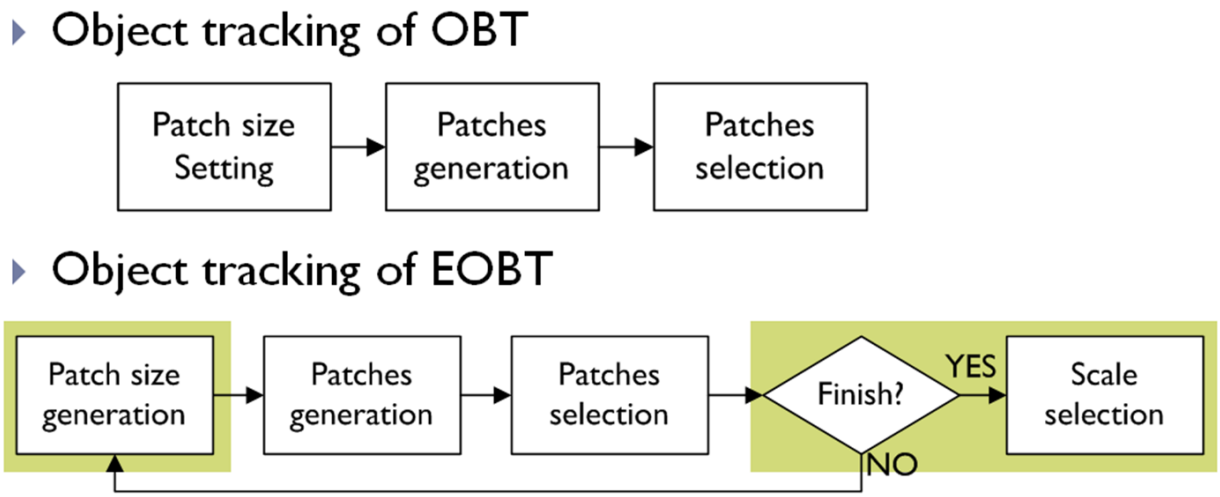


Figure 4.6: Revised tracking process of EOBT.

Normal scale adjustment is used on EOBT as Figure 4.5 shows. Multiple scales can be chosen by tracker's confidence value. The higher confidence, the more likely that guessing is correct. This is still a crucial problem since tracking accuracy is sometimes not so correct to judge object size. Once tracking accuracy is stable enough, tracker size could be stabilized. Hence, adding scalability on EOBT also exams stability and credibility of EOBT.

Since adding scalability to EOBT only revised tracking process of the overall system, we focus on this division to illustrate our revision. See Figure 4.6. Tracking process in OBT consists of four parts. First, set patch size the same as tracker size. Next, according to search region size and overlapped percentage, generate each patch one after another. These patches are selected by tracker. The one with highest confidence is object's new position. After object's new position is settled down, the tracker is then updated based on appearance of this new position. On the other hand, tracking process of EOBT modifies patch size setting and adds scale selection. In patch size generation, generate multiple scales for tracker. Each scale is evaluated and produced respective confidences. These confidences are used to estimate new object size in scale selection. Finally, classifier adaptation make tracker adapts to new appearance with the chosen scale.

4.3 Enhanced OBT with Lifetimer

If object is temporary occluded, conventional tracker exits and regards that tracked object is lost. However, in practical applications, there is no endless trail for real condition. Tracker has to inform succeeding processes that tracked object is lost or just temporary occluded. We consider that, the higher the number of successful tracking means the object is tracked tightly. Chances are that, abruptly zero confidence may be viewed as temporary occlusion after several successful tracking. Hence, we design a new mechanism for EOBT called *lifetimer* to make judgements. This mechanism is mainly designed to distinguish temporary occlusion from disappearance and can be implemented using different methods. Here, we use sigmoid function for example, because we should refer to the total number of its successful tracking. Sigmoid function, which is defined in Background, is rapid increasing or decreasing around the origin. This property answers our needs.

In Figure 4.7, the x-axis is tracker's lifetime. We use y-axis value to decide when disappearance occurred. Initial lifetime could be set any value, here we set it 0 for example. When tracking activated, every track returns a confidence value. In original design in OBT, once returned con-

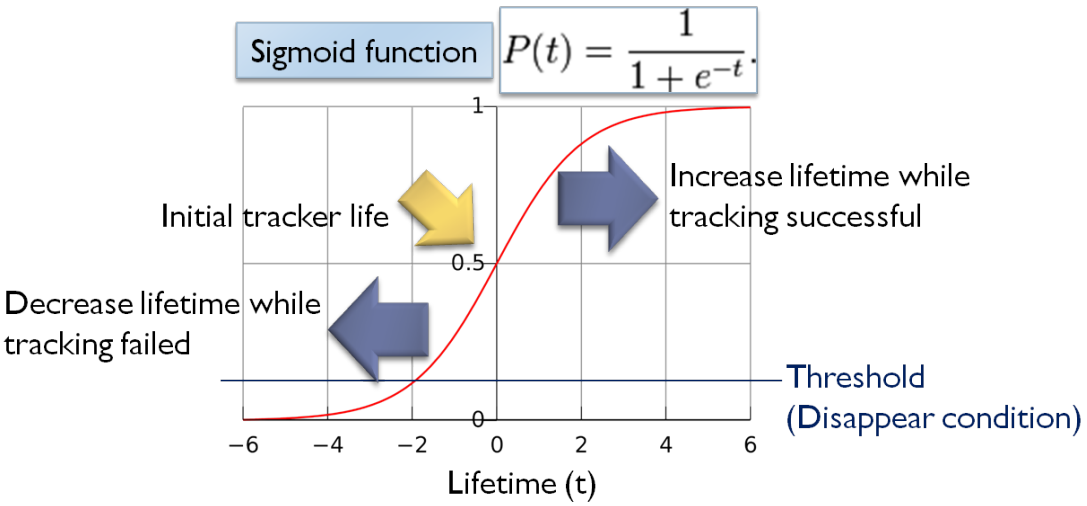


Figure 4.7: EOBT implemented with sigmoid function.

fidence is 0, the tracker is recognized as lost. However, in our design, when confidence is more than 0, tracker's lifetime increases. When confidence is stuck at 0, tracker's lifetime starts to decrease. Also, in case that the tracker has retained successful tracking for a long time and suddenly encountered disappearance, tracker needs to waste lots of time decreasing lifetime. The upmost value of lifetime can be set. Through lifetime design, tracker has authority to decide when to give up.

Since EOBT has only changed on disappearance judgement of OBT, we discuss those modifications. See Figure 4.8. In OBT, tracking is terminated if the tracker is lost or all frames are carried out. While in EOBT, opportunity is preserved for tracker. First, setting parameters like initial lifetime or disappear threshold. Then, lifetime is adjusted according to confidence of each track. Also, in each track, disappearance condition is examined, as shown in Figure 4.8.

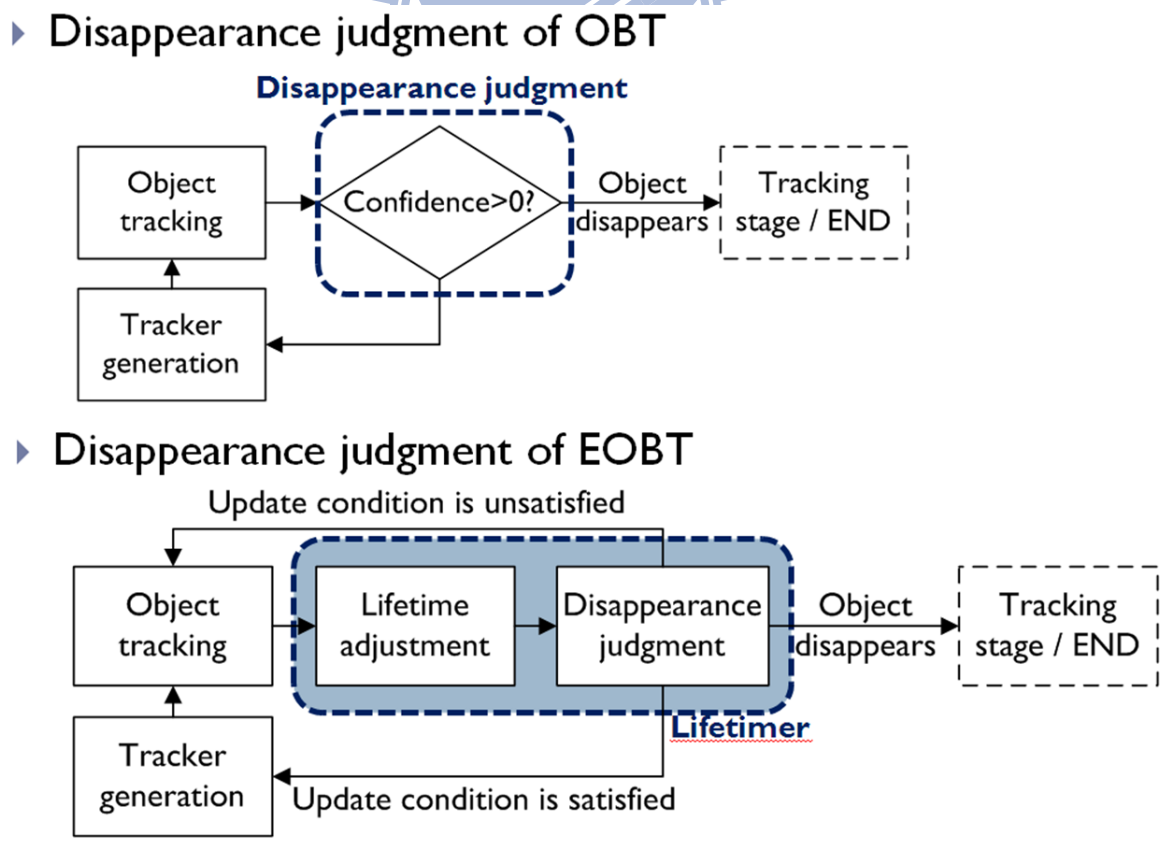


Figure 4.8: Disappearance judgement of EOBT

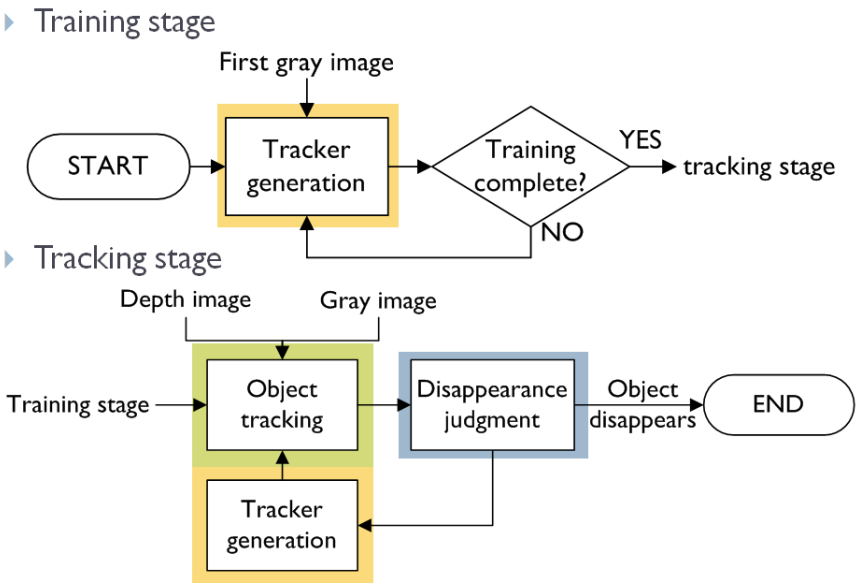


Figure 4.9: System overview of EOBT

To sum up, we have modified on several parts of OBT. See Figure 4.9 for better illustration. First, EOBT enhances OBT with depth image, which modifies 'Tracker generation' in system. Second, EOBT adds scalability on OBT using multiple scales, which modifies 'Tracking' part. Third, EOBT with lifetimer modifies disappearance condition to decide when should give up. These three mechanisms enhance original OBT in different parts, and the most important, we think these enhancements increase object integrity. We design three experiments to exam our proposal on next chapter. In discussion chapter, we discuss our tracking method with others and give a vivid view on object tracking.

Chapter 5

Experiments

In this chapter, we propose our evaluation method and use it to verify our proposal. First, preliminary gives an overview on evaluation method and implementation details. Next, three different experiments are carried out to test our proposal. Finally, summary is given in the last section.

5.1 Preliminary

In this section, we point out two drawbacks of conventional evaluation method and propose new evaluation method. We use proposed evaluation to judge experimental results. Also, implementation and assumption are mentioned in this section.

5.1.1 Evaluation Method

Recently, in several conference papers and journals, position error is a popular evaluation method in object tracking. It is easy to implement, but less accuracy and credibility. Since ground truth is labelled by human, variations of human labelling inevitably exist. If the variation of examined tracking method is more than those of human labelling, it is still an applicable evaluation. However, since these days tracking accuracy is promoted to certain level, chances are that variations of human labelling may influence accuracy judgement. Another issue is that, although better accuracy is shown on position error plot, the tracker already loses its focus. For example, see Figure 5.1. Tracked object is the 'coke can' held by hand. In position error plot,

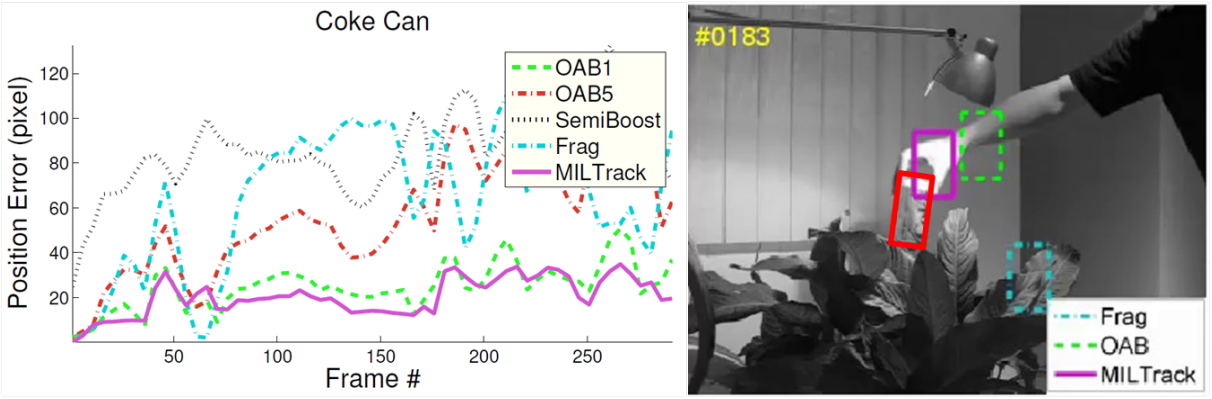


Figure 5.1: Wrong target problem of position error. Frame 183 in 'Coke Can' video. Adopted from [18]

MILTrack remains the best at frame #183. Nevertheless, the tracker has lost its focus and drifted to the wrist.

We think that tracked object pixels should be used to judge tracking results. These pixels represent area tracked object projects on to. If tracker has partially caught tracked object, it means the tracker is not lost. We illustrate this concept in Figure 5.2. There are two factors to judge tracking results. One factor is obviously, the percentage of tracked object in tracker, named RIO. This factor reflects the amount of tracked object area that tracker put attention on it. In addition, since scalable tracking method changes tracker size, percentage of tracked object in

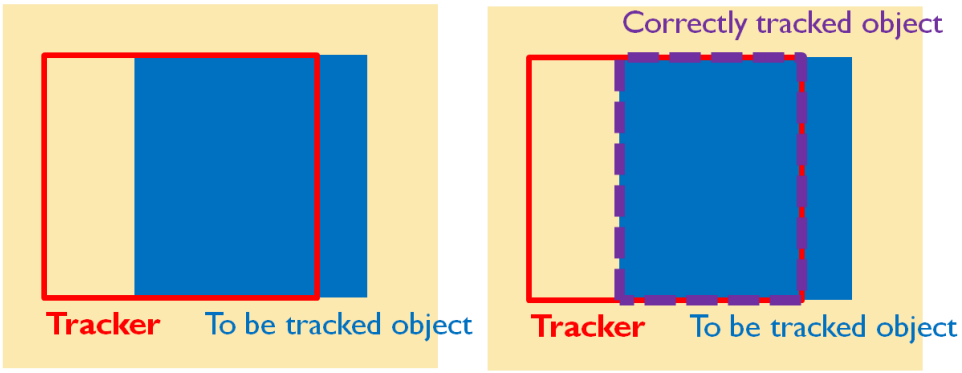


Figure 5.2: New evaluation method.

(a) (b) (c)

Figure 5.3: Example of ground truth construction.

tracker should be took into consideration. This is the purpose of our second factor, RIT. These two factors can be expressed using following equations.

$$RIO = \frac{\text{Pixels of correctly tracked object}}{\text{Total pixels of target object}} \tag{5.1}$$

$$RIT = \frac{\text{Pixels of correctly tracked object}}{\text{Pixels of tracker area}} \tag{5.2}$$

Proposed evaluation method may have problem on feasibility. Fortunately, thanks to the invention of Kinect, counting object total pixels becomes feasible. See Figure 5.3. Figure 5.3 (a) and 5.3(b) are images gathered by Kinect. Figure 5.3(c) is intercepted and binarized from depth image. Additional retouching is needed but not laborious. We use Figure 5.3(c) as ground truth for accuracy judgement.

5.1.2 Implementation

Here we list our experimental environment as follows.

1. PC

   (a) CPU: Intel®Core 2 DuoE7500 2.93 GHz

(b) RAM: 1.96 GB

2. Images from Microsoft®XBOX 360 Kinect

   (a) Color image: 640×480 resolution

   (b) Depth image: normalized depth values to 0-255

3. Libraries

   (a) OpenNI ver. 1.1.0.41 for image capture

   (b) OpenCV ver. 2.1 for image processing

5.1.3 Assumption

In our experiments, since features are generated randomly, every experiment is conducted 100 times and comparison is on their average performance. Our analysis method is different from Babenko's research [18]. Their comparison is based on the same features and only learning algorithm have been changed. Since stability of tracking methods is also one of our concerns, we choose statistical analysis. In experiments, number of weak classifiers and base classifiers are fixed. We list parameters of OBT and EOBT as follows.

- Number of base classifiers: 50
- Number of weak classifiers: 250
- Four parameters for Kalman filter using Gaussian distribution:

  $\mu_P = 500$; $\sigma_P = 0.0005$; $\mu_Q = 500$; $\sigma_Q = 0.0005$.

- Search factor: 4.0

Self-defined parameters are,

1. Tracker generation

   (a) Truncation method and its parameters

   For example, if fixed boundary is used, upper bound and lower bound value should be settled.

   (b) Background color

   This parameter is for tracking dark objects. Using light background color for dark objects is more appropriate.

2. Object tracking

   (a) Number of scales

   Set for number of scales.

   (b) Multiplying factor

   Multiplying factor of different scales according to previous tracker size.

3. Disappearance judgement

   (a) Initial lifetime

   (b) Increment

   Each time when tracking success, increase increment of lifetime.

   (c) Decrement

   Each time when tracking lost, decrease decrement of lifetime.

   (d) Disappearance threshold

   Threshold value for signifying tracked object disappear.

Especially, OMIL uses different weak classifier updating method. In our experiment, their updating method is remained. We briefly describe two different updating methods here. OMIL

uses,

$$\mu_t = \gamma\mu_{t-1} + (1-\gamma)\frac{1}{n}\sum_{i|y_i=1} f_k(x_i) \tag{5.3}$$

and

$$\sigma_t = \gamma\sigma_{t-1} + (1-\gamma)\sqrt{\frac{1}{n}\sum_{i|y_i=1}\big(f_k(x_i)-\mu_t\big)^2}. \tag{5.4}$$

with Bayes rule to determine threshold of Haar-like features. On the other hand, OBT and EOBT use more complex updating method. State space model and Kalman filter is used. State space is,

$$\mu_t = \mu_{t-1} + \nu_t, \tag{5.5}$$

$$\sigma_t^2 = \sigma_{t-1}^2 + \nu_t. \tag{5.6}$$

Parameters of Kalman filter are determined by,

$$K_t = P_{t-1}/(P_{t-1}+R), \tag{5.7}$$

$$\mu_t = K_t f_j(x) + (1-K_t)\mu_{t-1}, \tag{5.8}$$

$$\sigma_t^2 = K_t\big(f_j(x)-\mu_t\big)^2 + (1-K_t)\sigma_{t-1}^2, \tag{5.9}$$

$$P_t = (1-K_t)P_{t-1}. \tag{5.10}$$

The hypothesis returned from weak classifier is defined as,

$$h_j^{weak}(x) = p_j \times sign\big(f_j(x)-\vartheta_j\big), \tag{5.11}$$

where the threshold ϑ_j and parity $p_j$are,

$$\vartheta_j = |\mu^+ + \mu^-|/2 \tag{5.12}$$

and,

$$p_j = sign(\mu^+ - \mu^-). \tag{5.13}$$

The μ^+ is for positive samples and μ^- for negative samples.

Totally, there are three experiments, drifting, scalability and temporary occlusion. In each experiment, OBT, OMIL and EOBT are verified. Due to limitations of OSSB and BSST, they are not examined in our experiment. However, a comprehensive description is given in next chapter.

5.2 Drifting

In this experiment, Objective is to exam that EOBT helps to eliminate drifting. The scenario of experimental video is that one person does translation and rotation in frame sequence. It contains 360 rotation and translation at the same time. If adaptability and drifting-resistance of tracking methods are insufficient, drifting may occur in frame sequences. There are some examples of experimental video in Figure 5.4. Especially, color of frontal view and rear view is different. Therefore, color histogram-based tracking method might get lost in this situation.

Experimental results are depicted in Figure 5.5. Compared with OBT, OMIL and EOBT have great performance. Especially, Object that EOBT has tracked almost maintains more than 80% and after frame #130, EOBT has better performance than OMIL. Since tracker size of

Figure 5.4: Experimental video for experiment on drifting.

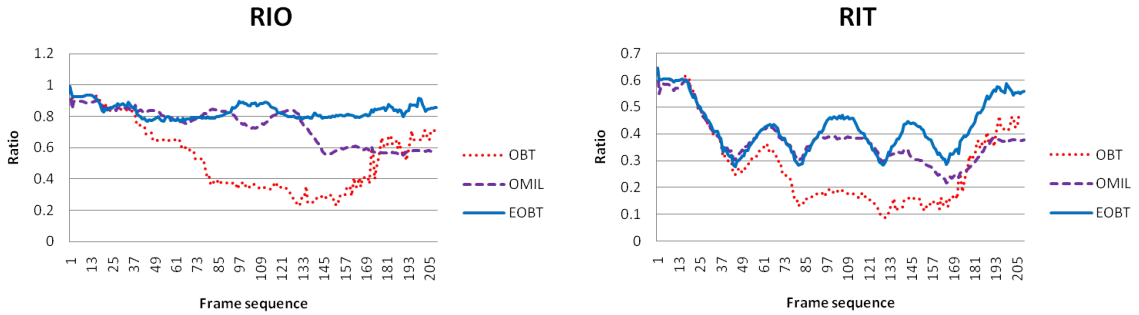


Figure 5.5: Experimental results on drifting.

all tracking methods remains the same in entire frame sequence, the tendency of RIO and RIT are similar. Difference is that, RIT curves become fluctuated. Reason of this phenomenon is caused by rotation of tracked person. Frontal view of tracked person is with the biggest area, which means more pixels are used to describe. On the other hand, lateral view is with the smallest area. Be aware that, in all of experiments, only frames have been tracked are took into consideration. Our purpose is to reveal actual tracking results. In this manner, the tracker's ability of judgements on missing condition is also correctly revealed from RIO and RIT curves.

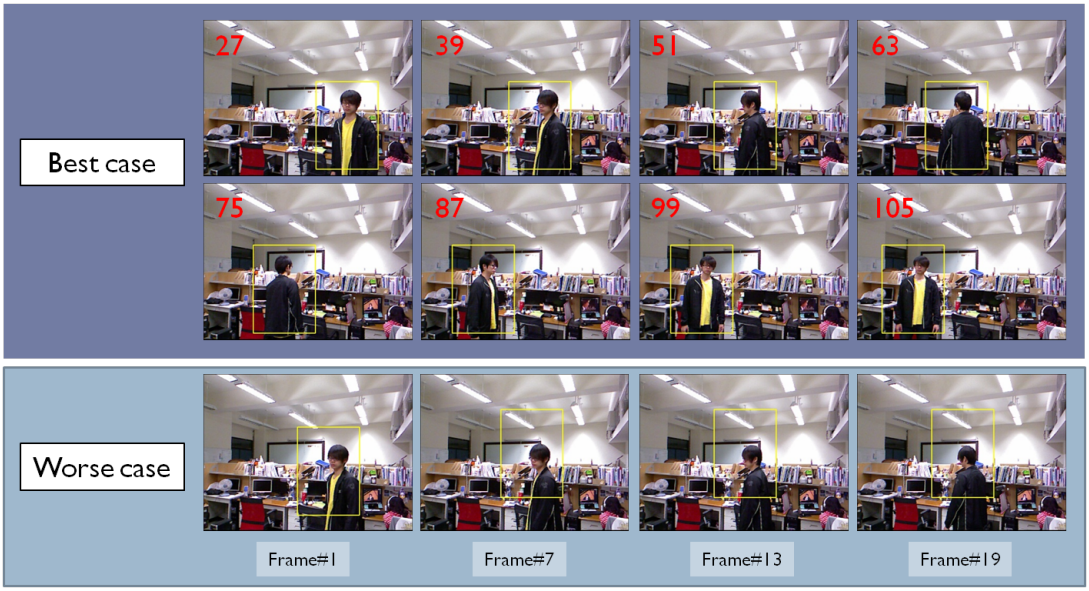


Figure 5.6: Best case and one of worse cases of our experimental results.

Some results are shown in Figure 5.6. Experimental results which are lost in half of frame sequence are not took here. In best case, focus of tracker is always on the tracked person. In worse case, however, focus in only put on upper part of the tracked person.

5.3 Scalability

In this experiment, tracking verification is on scalability. This verification is divided into two parts. First, tracked person moves from far to near. Second, this experimental video is played backward, so that different object motion can be evaluated. Thus, tracking ability on scalability is examined in two manners. One is from far to near. The other is from near to far. Figure 5.13 shows some example of experimental video. In addition, because OMIL does not adapt scale variation, we only compare EOBT with OBT here.

5.3.1 Moving Forwards

Experimental results are shown in Figure 5.8. From frame #1 to #3 in RIO, a drop on performance has occurred. This is caused by severe rotation of tracked person. Also, from frame #25 to #27 in RIT, performance is decreasing because of fragmented depth information. See Figure 5.9. In entire frame sequence, performance of EOBT is better than that of OBT. Especially, RIT has revealed that, in varied tracker size, EOBT has remained stable around 0.7.

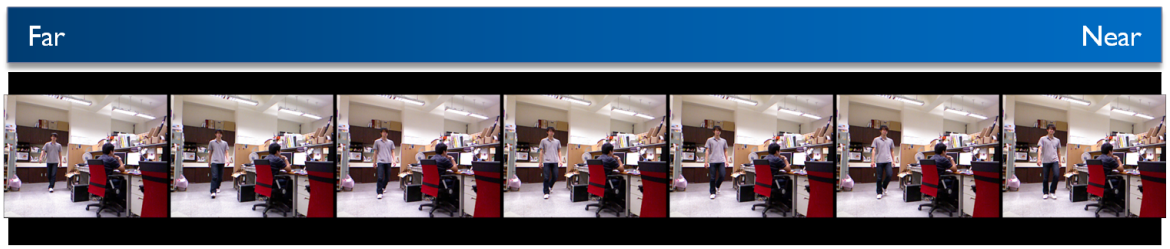


Figure 5.7: Experimental video for experiment on scales.

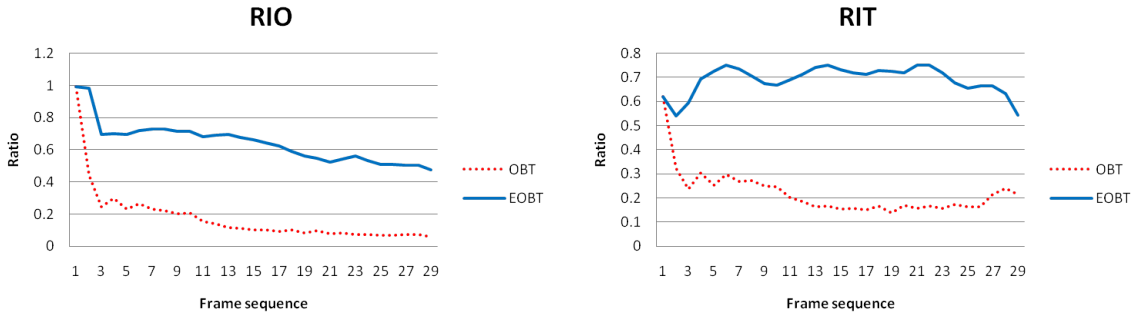


Figure 5.8: Experimental results on scalability. (From far to near)

Figure 5.9: Problems caused performance drops.

This means that tracker size adjustment catches up with moving speed of tracked object. If tracker size could not adjust in time, RIT curve becomes a descending curve. This situation is not happened to OBT since most trails of OBT is lost.

Figure 5.10 shows some of our experimental results. In best case, tracker size is catching up with tracked person. Be aware that these are consecutive frames from left to right and from first row to second row. In worse case, typical shrinking problem occurred just as tracking by Mean Shift [30]. This problem is caused by the distribution of random features. If representative features are gathered around at clothes of tracked person, smaller tracker gets higher confidence. We discuss this part in next chapter.

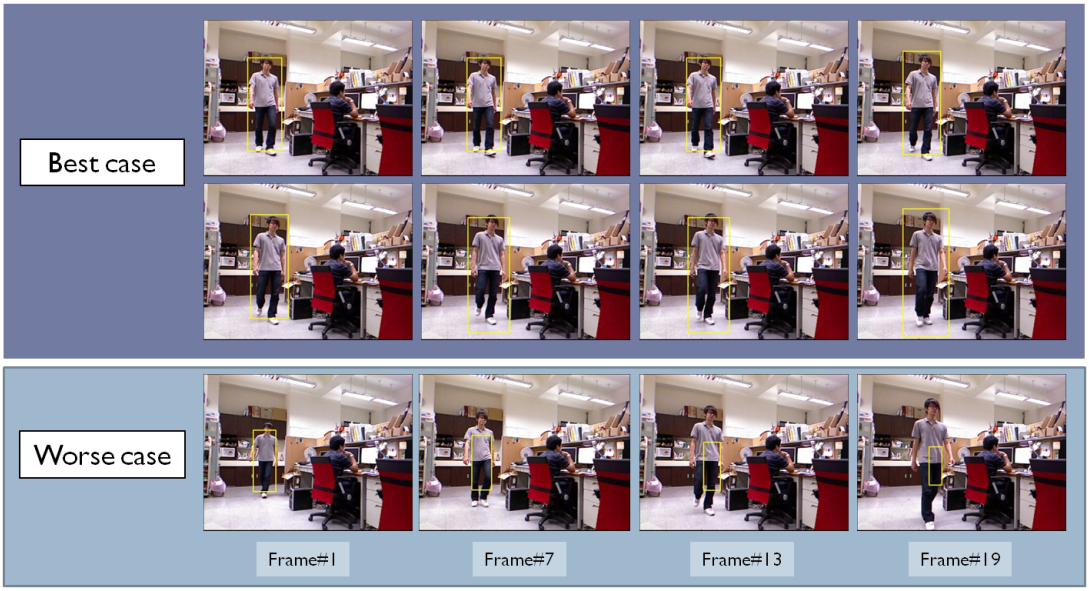


Figure 5.10: Best case and one of worse cases of our experimental results.(From far to near)

5.3.2 Moving Backwards

The experimental video is played backward to verify scalability of EOBT in different manner. Similar results are shown in Figure 5.11. Difference is that, previous mentioned descending curve has appeared here. Because of diminished area of tracked person, the part caught by OBT is decreased. If adjustment of tracker size caught up with tracked person, curves in both RIO and RIT remain horizontal. See EOBT curve in Figure 5.11. The ratio in RIO of EOBT remains between 0.8 and 0.9 from frame #2 to #25. The ratio in RIT decreases at first, but remain stable

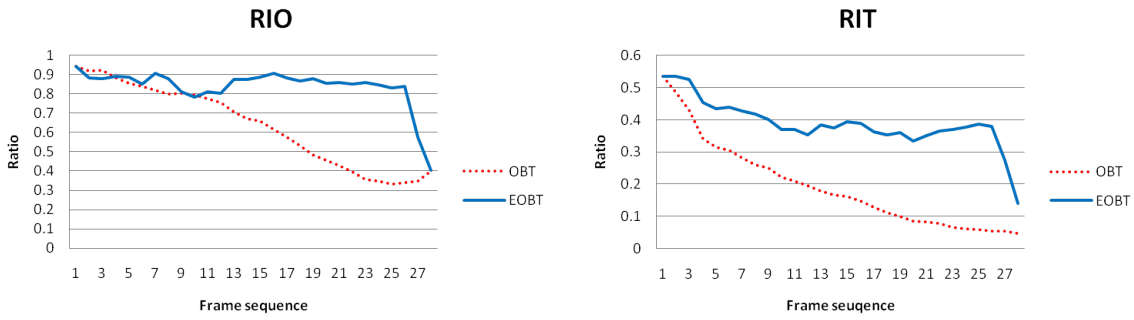


Figure 5.11: Experimental results on scalability.(From near to far)

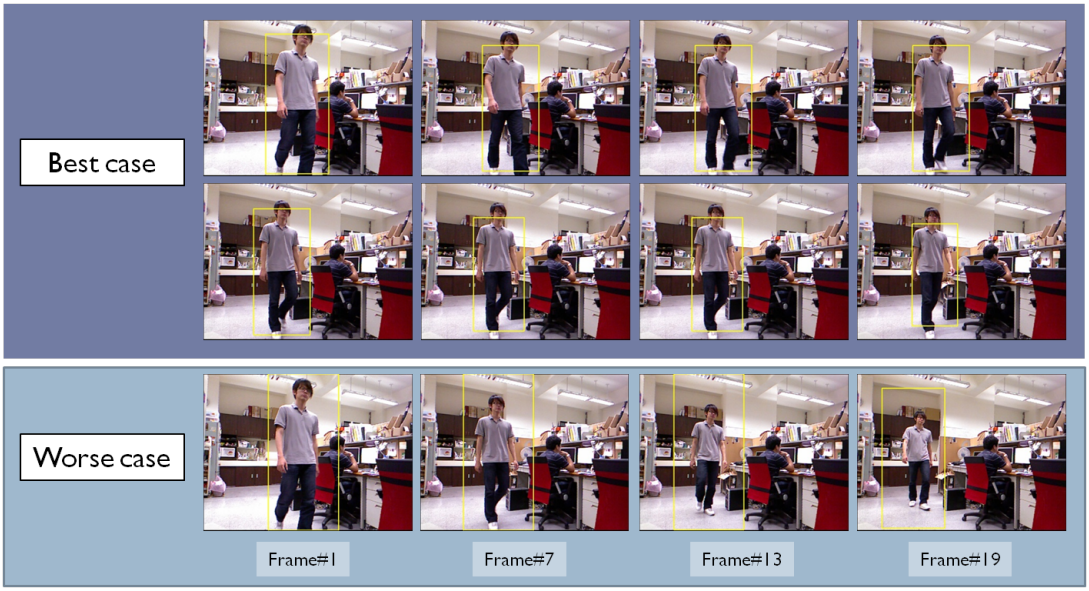


Figure 5.12: Best case and one of worse cases of our experimental results. (From near to far)

from frame #9 to #25. Drops of EOBT in both RIO and RIT are caused by severe rotation of tracked person, which has been mentioned before.

Experimental results are shown in Figure 5.12. The best case consists of consecutive frames, but worse case is not. In best case, the tracker has caught up with tracked person. In worse case, however, adjustment of tracker size is slower than the speed of variation of tracked target.

5.4 Temporary Occlusion

The third experiment verifies tracking methods on temporary occlusion. If object is occluded and tracker does not detect, the tracker then adapt on wrong target. Experimental video is designed as follows. Two people dressed clothes with similar color. They walk towards each other. The tracked person is the far one. When they pass each other, the tracked person is then severely occluded. Usually, tracker gets confused in this situation. Without any limitation, drifting occurred.

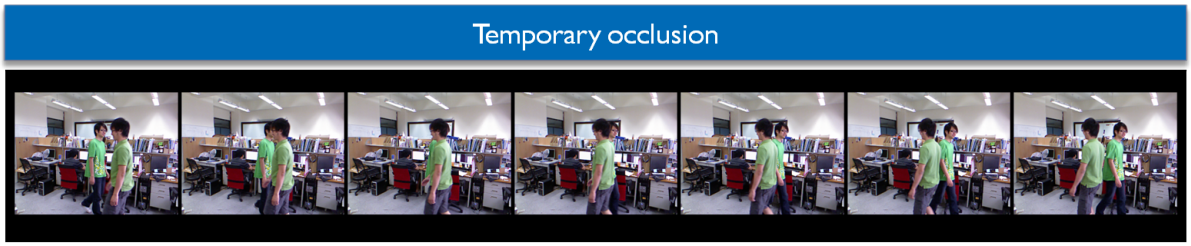


Figure 5.13: Experimental video for experiment on temporary occlusion.

Figure 5.14 shows experimental results. Top row shows several critical frames in video. OMIL and EOBT have better performance than OBT. From frame #62 to #67, the tracked person makes a relative huge stride. OMIL curves in both RIO and RIT have dropped abruptly. Performance of EOBT only decreases a little. From frame #77 to #83, the tracked person is occluded by the others. OMIL curves have dropped to zero in both RIO and RIT. On the other hand, EOBT curves only dropped at frame #80 and recovered after temporary occlusion. However, occlusion has destroyed tracker's inner object appearance model. Hence, some trails of EOBT have drifted away. This is the reason that EOBT curves decrease to zero after temporary occlusion. However, in our experiment, some EOBT does pass this situation. See Figure 5.15.

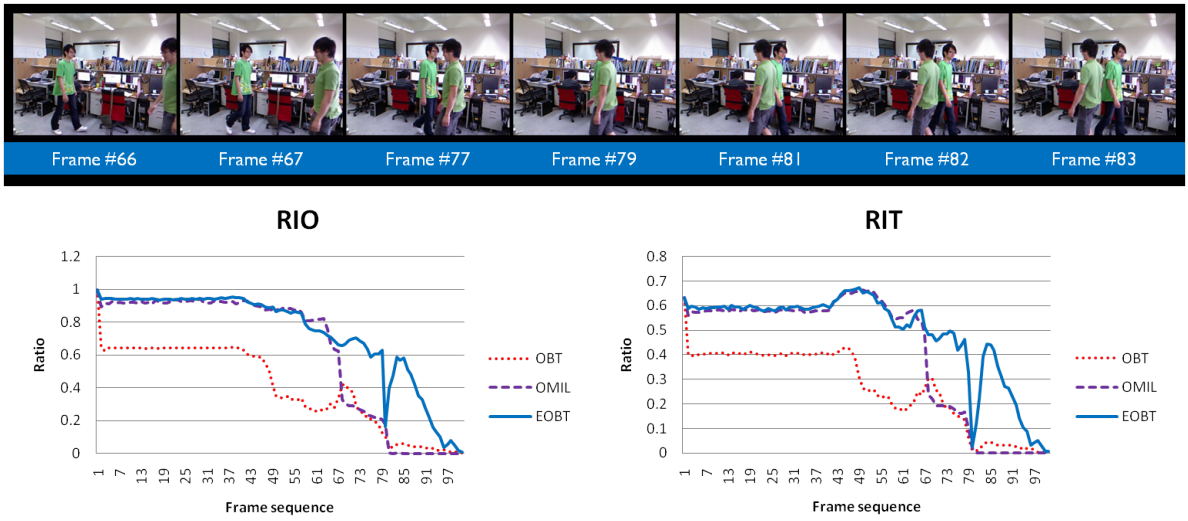


Figure 5.14: Experimental results on scalability.(From near to far)

Figure 5.15: Best case and one of worse cases of our experimental results.

In best case, the tracker focuses on correct person. While in temporary occlusion, tracking and updating are stopped. After that, tracker appears again. Although, the left image in second row of best case is not correct, however, after occlusion the tracker returns to correct person again. In worse case, since the appearance model is destroyed, the tracker drifts away.

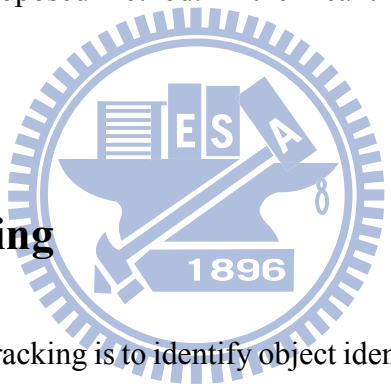
5.5 Summary

Experiments have examined tracking methods in three different aspects. First, rotation and translation cause drifting. Second, distance variation between camera and tracked object causes drifting. Third, drifting is caused by temporary occlusion. Experiments have shown that EOBT has better drifting-resistance ability than OBT and OMIL. Especially, EOBT eliminates drifting probability because it exploits depth information, which lowers background interference. In fact, there are countless situations might cause drifting. We try to clarify these situations and make a vivid understanding about object tracking in next chapter.

Chapter 6

Discussion

In this chapter, object tracking and tracking methods are illustrated in details. Definition of object tracking is always ambiguous, because it is highly coupled with object detection and object recognition. We have made a new definition according to proposed evaluation method. Also, tracking methods are discussed and difficulties are pointed out. We indicate contributions of related works and proposed method. In the meantime, roadmap of future work is also illustrated.



6.1 Object Tracking

The objective of object tracking is to identify object identity in the form of short-term memory. When one object comes in, visual system puts attention on it and perceives it as an independent identity without any recall. On the other hand, object recognition is much like comparing object with its long-term memory. To recognize it means to find the same object in memory. Object detection is a pre-process of object recognition. The understandings of object detection and recognition are in different levels. Applications like face detection find out where face is. However, detection does not tell whose face is. The concept it possess is only *face*.

Similar concept has been identified by Yang in [31]. Though short-term or long-term memory seems only applicable for human, our main focus is on acquisition of object identity without any recalls from stored datum. Challenge is that, one object consists of many appearances,

which is regarded as manifold. Existing tracking methods assume that changes of appearances are slow and continuous. They use adaptive appearance model to get through this challenge. Problem is, when using adaptive appearance model without any knowledge of real appearances of objects, drifting occurs. Similar problem has been addressed since 1996 in [32]. Research on manifold is still in progress. In fact, we consider that this challenge is tightly connected with semi-supervised learning [25]. Since semi-supervised learning is trying to bridge the gap between known data and unknown data, the maximum expandable knowledge of known data to unknown data is one of the submanifolds.

See Figure 6.1. $C_{k1}, C_{k2}...C_{k6}$ are submanifolds of M_k. One manifold can be regarded as the set of all appearances of a single object. In figure, **I** is the image obtained from camera. The challenge can be interpreted as to construct M_k. For every incoming image **I**, comparison is made by estimating the distance between **I** and manifold M_k, as $d_H(M_k, I)$ shown in figure. If M_k could be constructed during tracking, drifting might not occur.

The ultimate goal of object tracking is that, both RIO and RIT are approaching to 1.0. Also, every tracking result should be repeatable and variations between every trail should be as small

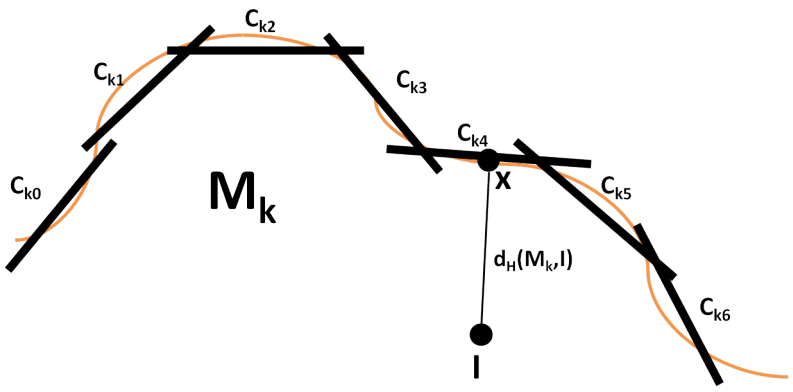


Figure 6.1: Concept of manifold. M_k is a specific manifold and C_{ki}s are submonifolds of M_k. I is some object. The distance d_H is used to calculate the similarity or likelihood that object I belongs to M_k.

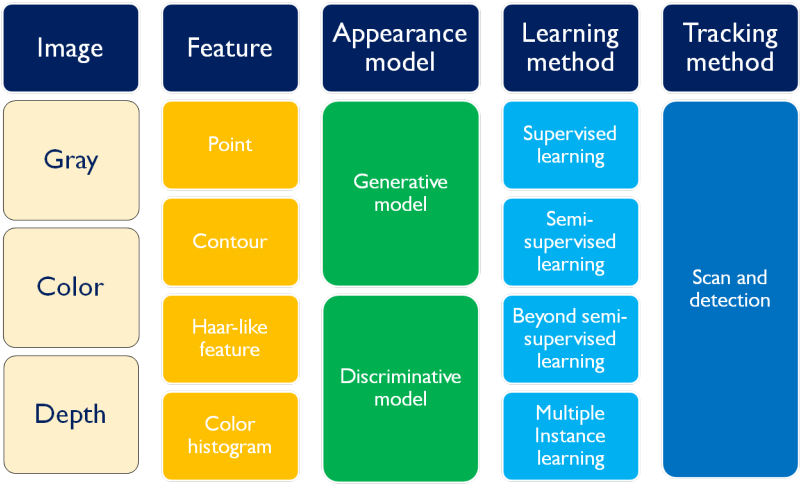


Figure 6.2: Integration of mentioned tracking methods

as possible. Repeatability and in-trail variations are described in next section.

6.2 Tracking Methods

In this section, illustration focuses on different aspects of tracking methods. Issues like methodology and stability are taken into our discussion.

6.2.1 Methodology

Tracking methods which first proposed by Grabner [13] can be divided into five segments as in Figure 6.2. In each segments, several researches have been conducted according to each part. Our discussion focus on these individual segments.

Image

Original design of OBT uses gray images as input. Gray images are also used as input for another three related researches, OSSB, BSST and OMIL. In our proposed method, depth image is used to distinguish object from background, lowering background interference at the

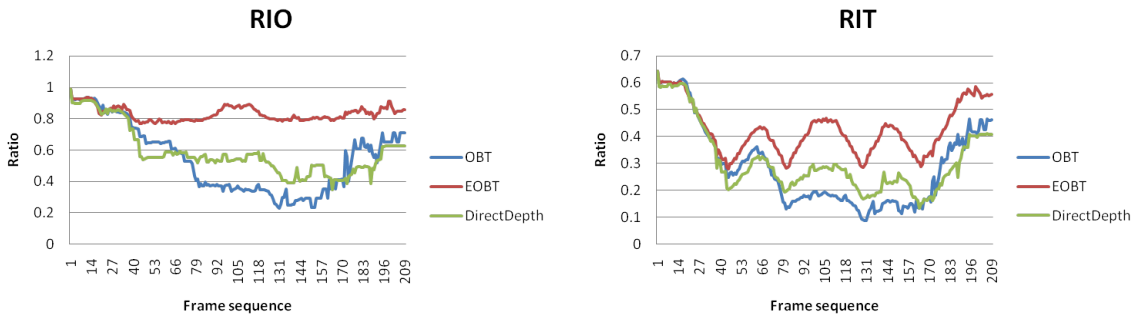


Figure 6.3: Experimental results on drifting.

same time. Also, we have directly used depth image and combined them with gray images and did experiments on drifting as stated before. Results are shown in Figure 6.3. It seems that direct use does not help OBT a lot. In fact, it is even worse after frame #170. We consider that, because Kinect use infrared ray to obtain depth information, reflected signal is not stable enough. We are not sure about the experimental results when using laser. Another potential source is color images. Color images possess three times information than gray image. They are more stable than depth images. We leave it as future work.

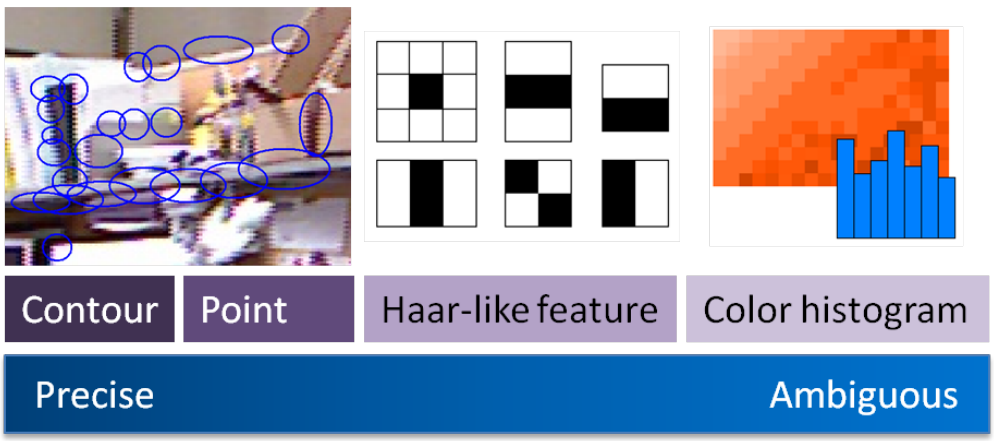


Figure 6.4: Ambiguity of each feature.

Feature

Related work and proposed method use Haar-like features. Since Viola and Jones exploit this feature in [21] and gain great success, Haar-like feature has been applied into many applications. The most amazing mechanism it offers is that, Haar-like feature is not too precise, but also preserves local characteristics. Color histogram, as mentioned in introduction, destroys orientation information. However, point and contour features are too precise, so they cannot endure large variations in physical environment. We believe that using Haar-like feature is part of reasons why tracking methods have abilities to tolerate variations. Figure 6.4 shows ambiguity of different features.

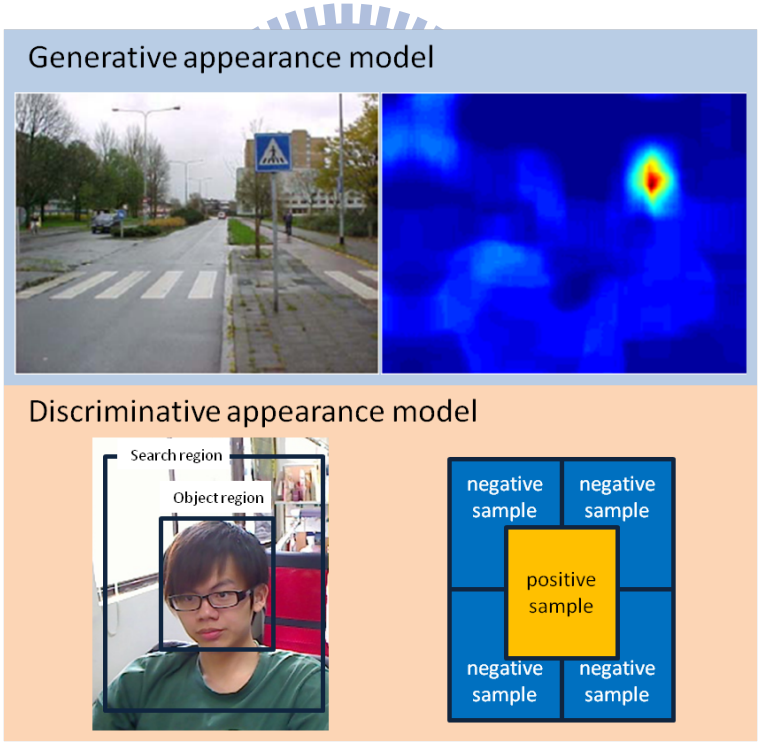


Figure 6.5: Two different appearance models.Generative model is adopted from [33]. Generative model is trained using object appearance directly. Discriminative model is trained using differences between objects and neighbouring regions.

Appearance Model

There are two different appearance models, generative appearance model and discriminative appearance model. Generative model collects data only from tracked object. On the other hand, discriminative model collects from not only tracked object, but surrounding background. Therefore, discriminative model can be regarded as combination of two generative models. One is from tracked object. The other is from surrounding background. Both of them are used in research. However, Yang has pointed out that,

> *many of the latter approaches have shown that training a model to separate the object from background via a discriminative classifier can often achieve superior results [31].*

Since more information is gathered, cost of processing time is paid. Trade-off between accuracy and time should be taken into consideration.

Learning Method

OBT, and also EOBT, are designed using supervised learning. They get through the manifold challenge by adapting new appearance. Since the reason why drifting is still inevitable has been mentioned, here we focus on semi-supervised learning and multiple instance learning.

Semi-supervised learning is a learning method that expands known data to those unknown. It should be useful in object tracking. However, it fails in all of our experiments. It is reasonable considering succeeding frames as unknown data. However, due to the characteristics of manifold, we consider that semi-supervised learning using similarity may not cross the gap between submanifolds. If tracking on similar appearances without changes on rotation and distance, semi-supervised learning method is stable and sufficient.

On the other hand, multiple instance learning modifies the sharp boundary between tracked object and surrounding background. Figure 6.6 shows the reason why OMIL can tolerate partial occlusion. We believe that, this is also the reason why OMIL is more stable than OBT. Next section illustrates tracking methods on stability. In fact, Godec et al. have mentioned the influence of label noise in [34]. They made initial tracker position slightly misaligned, and drifting occurred. However, label noise problem seems reduced in OMIL. Please refer to their website[1] and see their experimental video on David indoor. Integrating MIL and semi-supervised learning might solve manifold problem.

Tracking Method

In all related works and proposed EOBT, they track without any motion model. However, it is time-consuming to search all position. Godec et al. have exploited particle filtering to improve

[1] http://vision.ucsd.edu/ bbabenko/project_miltrack.shtml

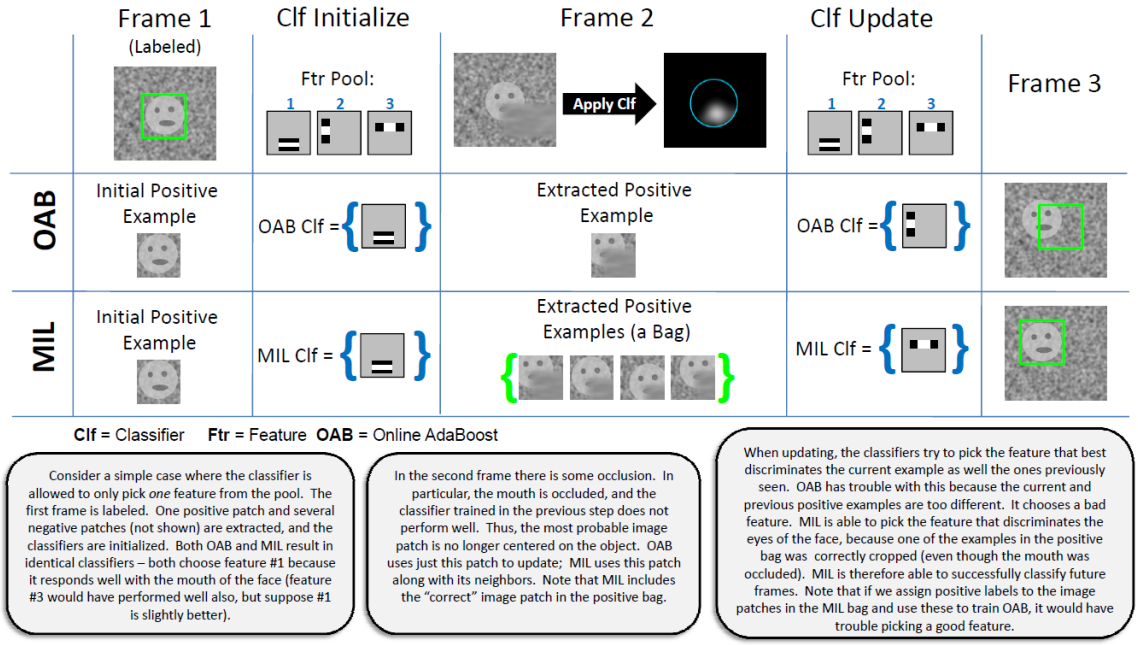
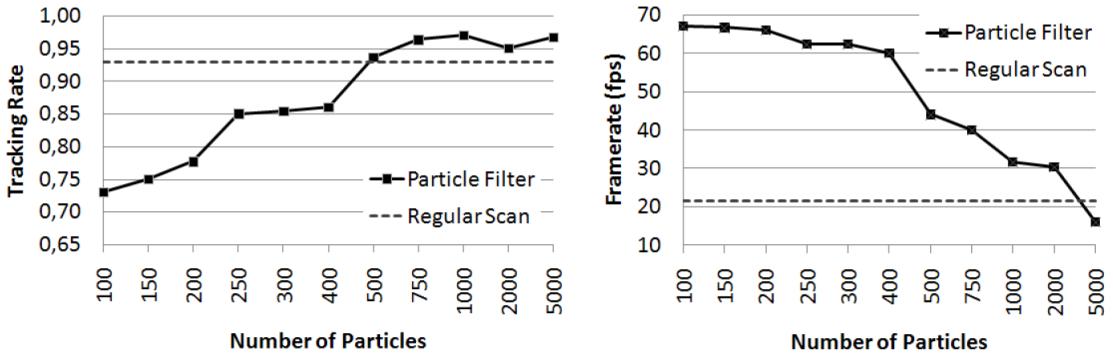


Figure 6.6: Solve occlusion problem using MIL [18]. Because MIL is trained from the concept of bag, each part of object appearance can be recognized. This helps a lot while object is occluded. Statements below the table describe how MIL works in detail.



(a) Tracking Rate

(b) Frame-rate

Figure 6.7: Experimental result on searching using particle filter. Adopted from [34].

search sampling. We adopted one of their experimental results in Figure 6.7. Their experiment shows that, number of particles between 500 and 3500 is applicable. Especially, frame rate is double when using 500 particles. Improve searching method also accelerates tracking performance.

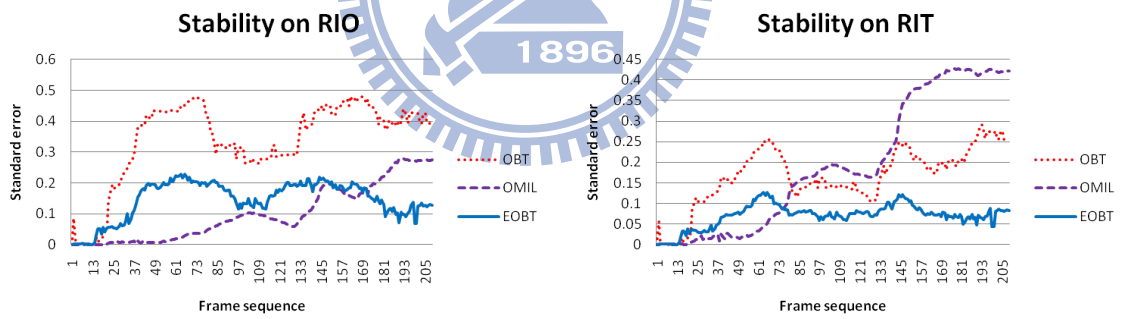


Figure 6.8: Stability on drifting.

6.2.2 Stability

Since Haar-like features are generated randomly, variation of tracking results should be one of considerations. Ultimate goal is the variation of tracking results approaches zero when using certain amount of features. We define this variation of tracking results as stability. It seems

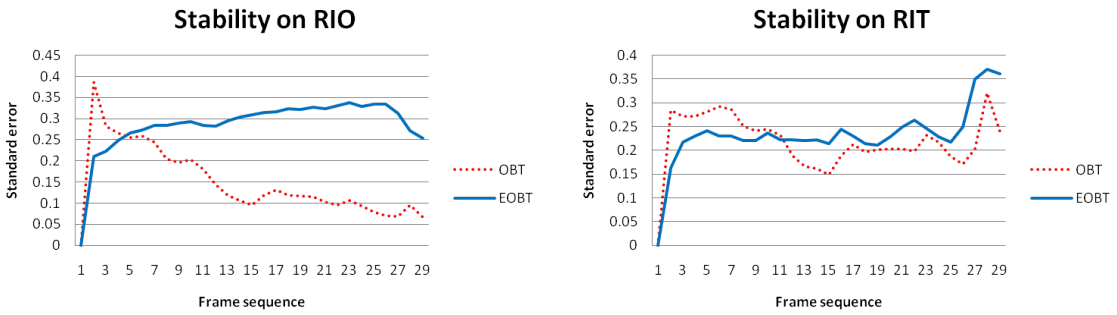


Figure 6.9: Stability on scalability.(Forward)

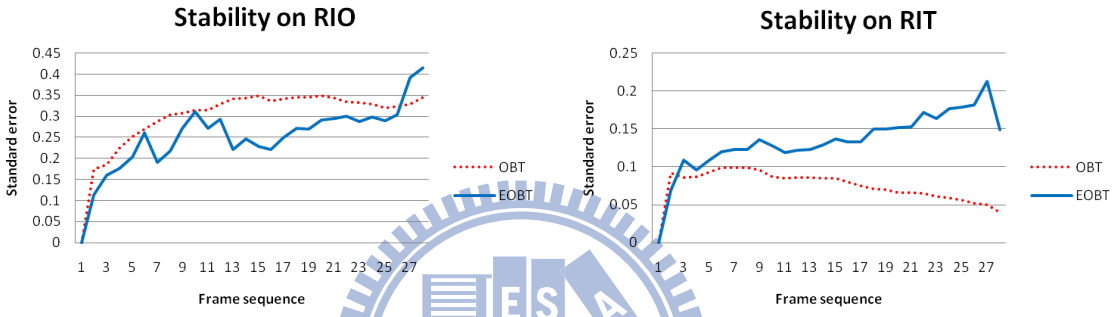


Figure 6.10: Stability on scalability.(Backward)

that stability is seldom addressed in research. Experimental results on stability are shown in Figure 6.8 to Figure 6.11. Especially, we use standard error to measure stability. Be aware that, we only compare results which have been tracked. Those lost results are not taken into account since we consider those are not reflected variations of tracking results on specific frame.

Experiment on drifting shows that, when tracked target is not lost, OMIL has the best stability. However, when drifting occurs, variation of OMIL rapidly increases. See Figure 6.8 in stability on RIT. When scalability is introduced into EOBT, stability decreases. Since multiple scales increase the degree of freedom, variation of tracking results is inevitably rising. See Figure 6.9 and Figure 6.10. When temporary occlusion is happened, stability of EOBT also decreases because occlusion destroys appearance models. It makes EOBT tracker unstable. Here, OBT and OMIL seem better than EOBT. However, their RIO and RIT performance has dropped

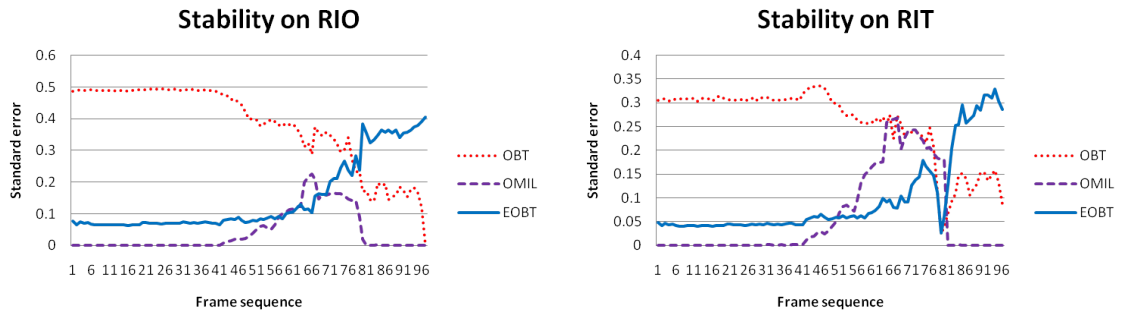


Figure 6.11: Stability on temporary occlusion.

to zero. Please refer to Figure 6.12. That means almost all trails are lost. EOBT with lifetimer has potential to get through temporary occlusion.

6.3 Summary

In this chapter, we have made our definition for object tracking. The key point is to identify *object identity*. Next, tracking methods using online boosting have been discussed from different aspects. Meanwhile, some future works are also proposed. Though stability is not good enough, experiments have shown that tracking using online boosting has excellent performance. We conclude all our research in next chapter.

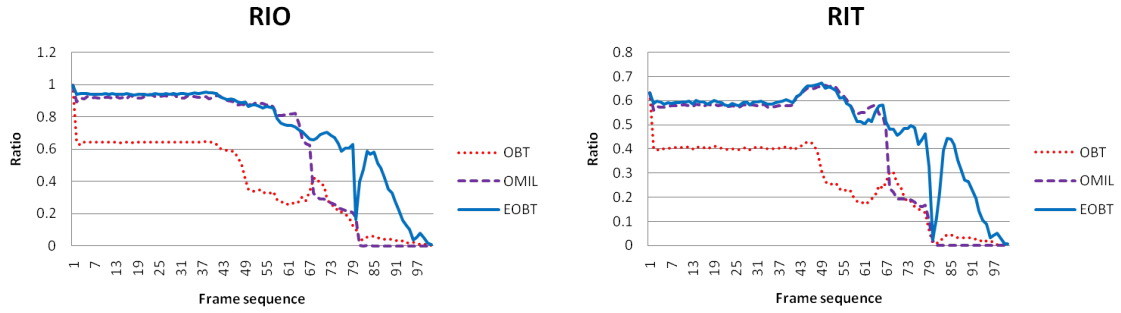


Figure 6.12: Experimental results on temporary occlusion.

Chapter 7

Conclusion and Future Work

In this thesis, we have proposed three mechanisms to enhance OBT in drifting resistance. First, we exploit depth information to lower background interference. Second, we introduce scalability for EOBT to solve drifting caused by distance variation. Third, we design a lifetimer so as to get through temporary occlusion. Also, we have proposed a new evaluation method to avoid wrong target problem in original evaluation method. New evaluation method is composed of two factors, RIO and RIT. RIO is a ratio to reveal the percentage of object that tracker has caught. RIT is used to reflect the area that tracker spends on tracking. Especially, RIT is used when tracker size is changeable.

All three of proposed mechanisms have been examined. Experimental results show that, EOBT does help to eliminate drifting probability. We also conduct research on stability, which is seldom addressed in related literature. Discussions on object tracking and tracking methods are described in previous chapter. Our future work will focus on solving manifold problem. Also, introducing parallel computing into tracking methods seems appropriate and may substantially accelerate performance. Improvements could be made in search sampling, and so on and so forth. Tracking using online boosting is still a developing research!

References

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm Computing Surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[2] D. Hoiem, A. Efros, and M. Hebert, "Putting objects in perspective," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 3--15, 2008.

[3] S. Nejhum, J. Ho, and M. Yang, "Visual tracking with histograms and articulating blocks," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1 -- 8.

[4] C. Tomasi and T. Kanade, *Detection and tracking of point features*. Citeseer, 1991.

[5] K. Cannons, "A review of visual tracking," Technical Report CSE-2008-07, York University, Department of Computer Science and Engineering, Tech. Rep., 2008.

[6] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15. Manchester, UK, 1988, p. 50.

[7] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91--110, 2004.

[8] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Springer Berlin / Heidelberg, 2006, vol. 3951, pp. 404 -- 417, 10.1007/11744023-32. [Online]. Available: http://dx.doi.org/10.1007/11744023_32

[9] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, pp. 321--331, 1988, 10.1007/BF00133570. [Online]. Available: http://dx.doi.org/10.1007/BF00133570

[10] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, pp. 61--79, 1997, 10.1023/A:1007979827043. [Online]. Available: http://dx.doi.org/10.1023/A:1007979827043

[11] Y. Ming-Hsuan, "Advances in visual tracking," Asian Conference on Computer Vision (ACCV), 2010.

[12] G. Welch and G. Bishop, "An introduction to the kalman filter," Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, Tech. Rep., 2006.

[13] H. Grabner and H. Bischof, "On-line boosting and vision," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 260 -- 267.

[14] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. BMVC*, vol. 1. Citeseer, 2006, pp. 47 -- 56.

[15] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance**," *Cognitive science*, vol. 11, no. 1, pp. 23--63, 1987.

[16] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Computer Vision – ECCV 2008*, ser. Lecture Notes in Computer Science, D. Forsyth, P. Torr, and A. Zisserman, Eds. Springer Berlin / Heidelberg, 2008, vol. 5302, pp. 234 -- 247, 10.1007/978-3-540-88682-2-19. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88682-2_19

[17] S. Stalder, H. Grabner, and L. Van Gool, "Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1409 -- 1416.

[18] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, june 2009, pp. 983 --990.

[19] F. Yoav and E. Robert, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119--139, 1997.

[20] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337--407, 2000.

[21] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137--154, 2004.

[22] N. Oza, "Online Ensemble Learning," Ph.D. dissertation, UNIVERSITY of CALIFORNIA, 2001.

[23] S. Nejhum, J. Ho, and M. Yang, "Online visual tracking with histograms and articulating blocks," *Computer Vision and Image Understanding*, vol. 114, no. 8, pp. 901--914, 2010.

[24] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *cvpr*. Published by the IEEE Computer Society, 2000, p. 2142.

[25] X. Zhu, "Semi-supervised learning literature survey," Computer Science, University of Wisconsin-Madison, Tech. Rep. 1530, 2008.

[26] P. Mallapragada, R. Jin, A. Jain, and Y. Liu, "Semiboost: Boosting for semi-supervised learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 11, pp. 2000--2014, 2009.

[27] C. Leistner, H. Grabner, and H. Bischof, "Semi-supervised boosting using visual similarity learning," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1 -- 8.

[28] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Advances in neural information processing systems*. Citeseer, 1998, pp. 570 -- 576.

[29] P. Viola, J. Platt, and C. Zhang, "Multiple instance boosting for object detection," *Advances in neural information processing systems*, vol. 18, p. 1417, 2006.

[30] R. Collins, "Mean-shift blob tracking through scale space," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, june 2003, pp. II -- 234--40 vol.2.

[31] M.-H. Yang and J. Ho, "Toward robust online visual tracking," in *Distributed Video Sensor Networks*, B. Bhanu, C. V. Ravishankar, A. K. Roy-Chowdhury, H. Aghajan, and D. Terzopoulos, Eds. Springer London, 2011, pp. 119 -- 136, 10.1007/978-0-85729-127-1--8. [Online]. Available: http://dx.doi.org/10.1007/978-0-85729-127-1--8

[32] P. Belhumeur and D. Kriegman, "What is the set of images of an object under all possible lighting conditions?" in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, jun 1996, pp. 270 --277.

[33] F. Porikli, "Integral histogram: a fast way to extract histograms in cartesian spaces," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, june 2005, pp. 829 -- 836 vol. 1.

[34] M. Godec, H. Grabner, C. Leistner, and H. Bischof, "Speeding up semi-supervised on-line boosting for tracking," 2010.

[35] A. Singh, R. Nowak, and X. Zhu, "Unlabeled data: Now it helps, now it doesn't," *Advances in Neural Information Processing Systems*, vol. 21, no. 2008, pp. 1513--1520, 2008.

Appendix A

Appendices

Theories which used in this paper are describe here in detail. Because online boosting is based on boosting, boosting is illustrated first, and then online boosting comes after. Next, introduce semi-supervised learning, which is used in one of our related works. Last but not least, we introduce multiple instance learning.

A.1 Boosting

Boosting is an ensemble learning algorithm. Its goal is to combine several weak classifiers into a strong classifier. Here, weak classifier means learning algorithm which is not correct enough. Strong classifier, on the other hand, makes accuracy classification. This is the reason why it called ensemble learning.

The main idea of boosting is to adjust weight distribution on samples. Boosting gives every training sample a corresponding weight. At each round of learning, sample weight is adjusted so that weak classifiers focus on those misclassified samples. There are several different versions of boosting, like AdaBoost, GentleBoost, and so on. We choose AdaBoost in [22] for better connection with online boosting. One may refer to [19] for detailed illustration in theories of AdaBoost. See Figure A.1

First, (x_1, y_1) is a pair of training sample. x_1 is a sample value and y_1 is its corresponding categories. L_b is any adopted weak learning algorithm and M is number of weak classifiers. In initial stage, sample weight is given for every sample as $^1/_N$. When learning starts, each weak

AdaBoost$(\{(x_1, y_1), \ldots, (x_N, y_N)\}, L_b, M)$

Initialize $D_1(n) = 1/N$ for all $n \in \{1, 2, \ldots, N\}$.

For $m = 1, 2, \ldots, M$:

$h_m = L_b(\{(x_1, y_1), \ldots, (x_N, y_N)\}, D_m)$.

Calculate the error of h_m : $\epsilon_m = \sum_{n:h_m(x_n) \neq y_n} D_m(n)$.

If $\epsilon_m \geq 1/2$ then,

set $M = m - 1$ and abort this loop.

Update distribution D_m:

$$D_{m+1}(n) = D_m(n) \times \begin{cases} \frac{1}{2(1-\epsilon_m)} & \text{if } h_m(x_n) = y_n \\ \frac{1}{2\epsilon_m} & \text{otherwise} \end{cases}$$

Output the final hypothesis:

$h_{fin}(x) = \text{argmax}_{y \in Y} \sum_{m:h_m(x)=y} log \frac{1-\epsilon_m}{\epsilon_m}$.

Figure A.1: AdaBoost algorithm [22].

classifier learns from all samples using learning algorithm L_b with the same weights. After learned from all samples, classification errors ϵ_m is summed. Classification error is number of misclassified samples. If more than half samples were misclassified, it means that this learning is worst than random guess and trained classifier is discarded. If classification error was less than $^1/_2$, sample weight is adjusted. Weights of those correctly classified samples are reduced. Their previous weight is multiplied by $\frac{1}{2(1-\epsilon_m)}$. On the other hand, weights of misclassified samples are raised and their weights are multiplied by $\frac{1}{2\epsilon_m}$. In this manner, boosting focus on 'hard' samples.

After M weak classifiers are finished learning, final strong classifier is linearly combined by weak classifiers. Here, hypothesis means prediction made by classifiers. The final hypothesis chooses the maximum value of combinations of weak hypotheses which voted for each category.

Here, if only two categories are in use, the final hypothesis is simpler as,

$$h_{fin}(x) = \sum_{m:h_m(x)=y} \log \frac{1-\epsilon_m}{\epsilon_m} \tag{A.1}$$

Be aware that, ϵ_m here is the hypothesis made by m weak classifier after learning.

Oza has explained why the combination factor of each weak hypothesis is

$$\sum_{m:h_m(x)=y} \log \frac{1-\epsilon_m}{\epsilon_m}. \tag{A.2}$$

We extract his deduction as follows. If Bayes optimal decision rule is used as learning algorithm and only two-category problem are dealt with, the reason we choose y_1 over y_2 is that,

$$P(Y = y_1|h_1(x), ..., h_M(x)) > P(Y = y_2|h_1(x), ..., h_M(x)). \tag{A.3}$$

According to Bayes rule, Eq. A.3 can be rewritten as,

$$\frac{P(Y = y_1)P(h_1(x), ..., h_M(x)|Y = y_1)}{P(h_1(x), ..., h_M(x))} > \frac{P(Y = y_2)P(h_1(x), ..., h_M(x)|Y = y_2)}{P(h_1(x), ..., h_M(x))}. \tag{A.4}$$

Since two denominators are same for all categories, we eliminate them together. Also, because $h_1(x), ..., h_M(x)$ are independent events, Eq. A.4 is the same as

$$P(Y = y_1) \prod_{m:h_m(x)\neq y_1} \epsilon_m \prod_{m:h_m(x)=y_1} (1-\epsilon_m) > P(Y = y_2) \prod_{m:h_m(x)\neq y_2} \epsilon_m \prod_{m:h_m(x)=y_2} (1-\epsilon_m). \tag{A.5}$$

Here, $h_m(x) \neq y_1$ is the same as $h_m(x) = y_2$ because there are only two categories. $P(Y = y_1)$ and $P(Y = y_2)$ are trivial since we can add a weak classifier which always predict y_1. Then, $P(Y = y_1)$ could be replaced by $1 - \epsilon_0$ and $P(Y = y_2)$ is replaced by ϵ_0. We get,

$$\prod_{m:h_m(x)=y_1} \frac{1-\epsilon_m}{\epsilon_m} > \prod_{m:h_m(x)=y_2} \frac{1-\epsilon_m}{\epsilon_m}. \tag{A.6}$$

Taking logarithms, we get,

$$\sum_{m:h_m(x)=y_1} \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right) > \sum_{m:h_m(x)=y_2} \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right). \tag{A.7}$$

Eq. A.7 tells the reason why weights of each weak classifier is set as

$$\sum_{m:h_m(x)=y} \log \frac{1-\epsilon_m}{\epsilon_m}. \tag{A.8}$$

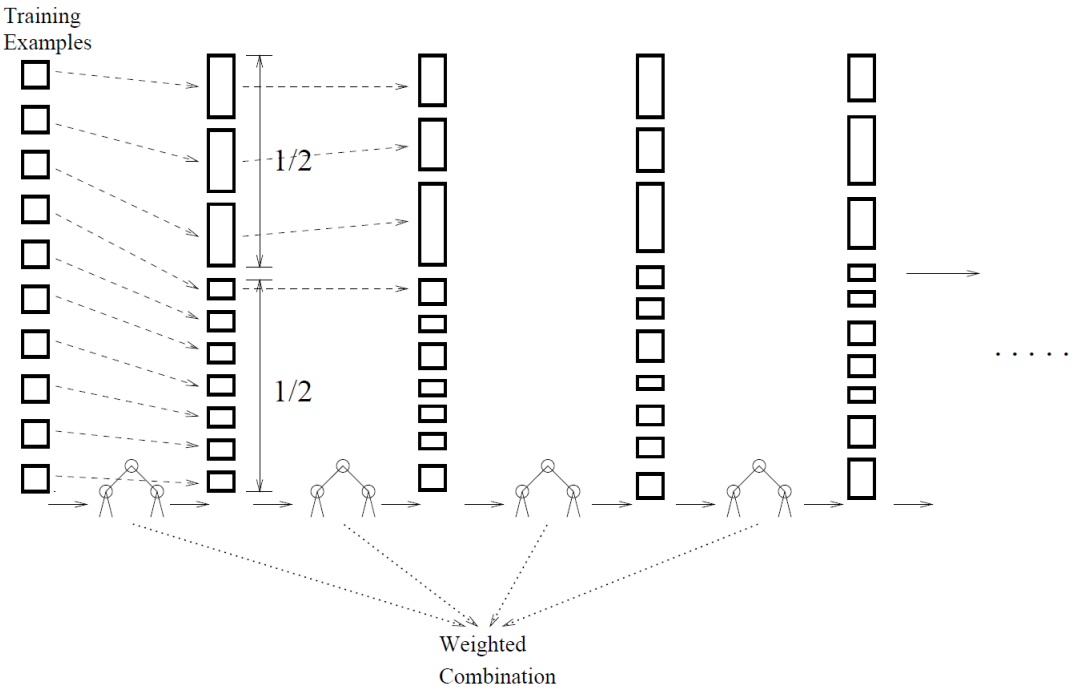


Figure A.2: AdaBoost in action [22].

Figure A.2 shows AdaBoost in action. The rectangle is represented as sample weight. At first, sample weights for each samples are the same. After learning by a weak classifier, sample weights are adjusted according to hypotheses made by this weak classifier. This process executes for several rounds until target classification error rate is reached.

AdaBoost is proved that the maximum error returned by strong classifier is bounded above by

$$2^M \prod_{m=1}^{M} \sqrt{\epsilon_m(1-\epsilon_m)}. \tag{A.9}$$

Please refer to [19] for details. Oza has pointed out that, according to several experiments, adding more weak classifiers even after classification error of final strong has reached zero, overfitting is not occurred and margin of samples continues to increase. Here margin of sam-

ples is the total weighted vote for correct category minus the total weighted vote for incorrect category. This experimental results and its excellent performance have made AdaBoost one of the most popular learning algorithms.

A.2 Online Boosting

Oza has made boosting from batch processing into online processing. Online processing means algorithm only learns each training sample once and discards it forever. To clarify, boosting in section A.1 is rename as batch boosting. Online processing introduced by Oza is named online boosting. To make batch boosting to become online, the concept of *lossless* should be mentioned beforehand. As defined in [22], a lossless online learning algorithm is an algorithm that returns a hypothesis identical to what its corresponding batch algorithm would return given the same training set. Several lossless online algorithms are decision trees, decision stumps and Naive Bayes classifiers. Using lossless algorithm is one of key points which maintains online boosting as good as batch boosting when training set is become huge.

See Figure A.3, algorithm of online boosting. In previous section, we name every learning algorithm as a weak classifier. Here, weak classifiers are regard as base of online boosting. So, weak classifiers are named as base model. In essence, they are the same. First of all, decide the amount of base models and set λ_m^{sc} and λ_m^{sw} as zero. λ_m^{sc} defines as the sums of weight of the currently classified examples, and λ_m^{sw} is of the currently misclassified examples. Initially, example weight, which is the sample weight mentioned before, is set to one. Next, every base model learns k times from the incoming example. Although k is set according to Poisson distribution, $k - Possion(\lambda)$, defined as,

$$f(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{A.10}$$

Initial conditions: For all $m \in \{1, 2, \ldots, M\}$, $\lambda_m^{sc} = 0, \lambda_m^{sw} = 0$.

OnlineBoosting($\mathbf{h}, L_o, (x, y)$)

Set the example's "weight" $\lambda = 1$.

For each base model h_m, ($m \in \{1, 2, \ldots, M\}$) in $\mathbf{h}$,

Set k according to $Poisson(\lambda_d)$.

Do k times

$h_m = L_o(h_m, (x, y))$

If $y = h_m(x)$

then

$\lambda_m^{sc} \longleftarrow \lambda_m^{sc} + \lambda$

$\epsilon_m \longleftarrow \frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}}$

$\lambda \longleftarrow \lambda \left(\frac{1}{2(1-\epsilon_m)} \right)$

else

$\lambda_m^{sw} \longleftarrow \lambda_m^{sw} + \lambda$

$\epsilon_m \longleftarrow \frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}}$

$\lambda \longleftarrow \lambda \left(\frac{1}{2\epsilon_m} \right)$

To classify new examples:

Return $h(x) = \text{argmax}_{c \in Y} \sum_{m: h_m(x) = c} log \frac{1-\epsilon_m}{\epsilon_m}$.

Figure A.3: Algorithm of online boosting [22]

practically we could set it as a constant. Then, verify this base model using current example. If it does classify correctly, add example weight, λ, to λ_m^{sc} and calculate the error of this base model by,

$$\epsilon_m = \frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}}. \tag{A.11}$$

The error calculation is the first difference compared with batch boosting. Another difference is example weight adjustment. In batch boosting, this is done according to the overall training set. While in online boosting, example weight adjustment is solely done according to error of one base model, which defined as,

$$\lambda = \lambda \left(\frac{1}{2(1 - \epsilon_m)} \right) \tag{A.12}$$

However, example weight adjustment is operated as batch boosting in that, for misclassified example, example weight is raised and for correctly classified example, example weight is reduced. The final strong classifier of online boosting is composed the same as its counterpart in batch boosting. For more details and theories, please refer to Oza's ph.D. thesis [22].

To better illustrate, see Figure A.4. In each row, every base model learns from only one example. In first row from left to right, training example is learned by first base model. After that, first base model returns an error according to its classification on this example. The error returned by first base model is used to adjust example weight. Altered rectangle size shows example weight adjustment. When all training set are learned, these base models are then weighted combined according to its error.

Oza has claimed that, using lossless online algorithm to create base models, the performances of online boosting is not far from batch boosting. This achievement is also the foothold

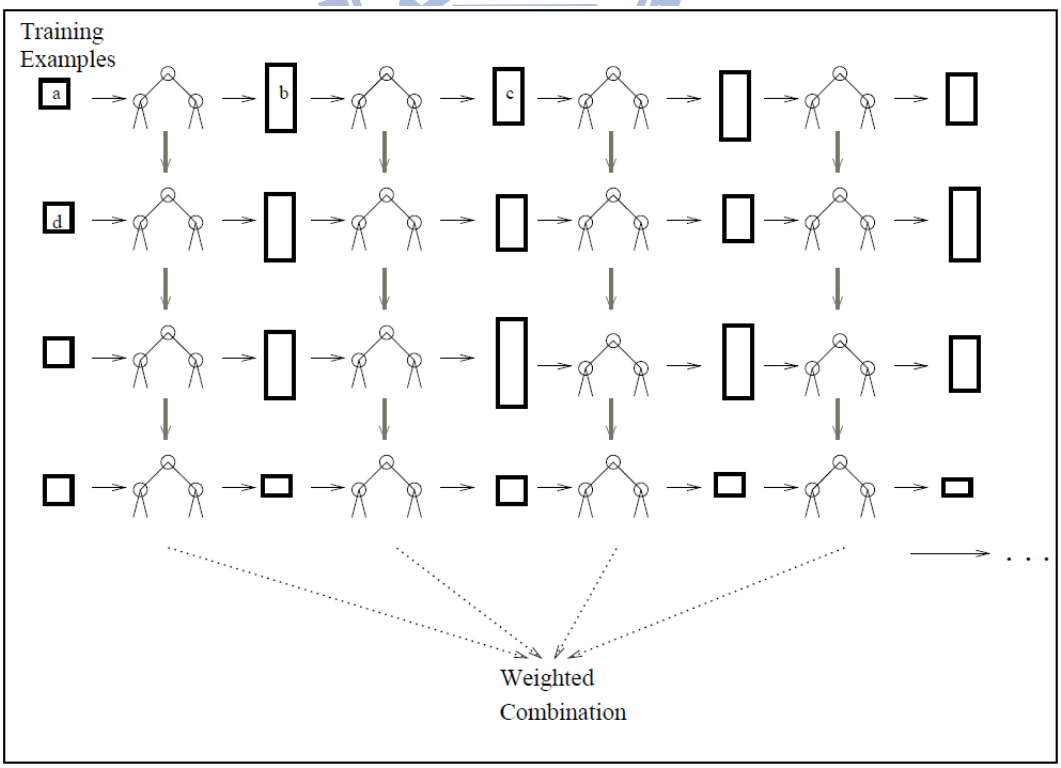


Figure A.4: Online boosting in action [22].

for successful applying online boosting to object tracking.

A.3 Semi-Supervised Learning

Semi-supervied learning exploits not only labeled data but also unlabeled data. Traditional learning methods use only labeled training data, which are called supervised learning. Here, labeled data means data is classified correctly by human. Sufficient labeled data is always a problem for machine learning. Semi-supervised learning bridge the gap between labeled and unlabeled data. There are many methods designed for semi-supervised learning, like EM with generative mixture models, self-training, co-training, etc. Xiaojin [25] has made a comprehensive survey on semi-supervised learning. Here since SemiBoost [26] is adopted by Grabner for tracking in [16], illustration focuses on SemiBoost. Especially, for better connection with tracking, we adopt [27].

In [20], boosting is regarded as adaptive logistic regression which minimize an exponential loss function on training data. Loss function could be viewed as objective function. Boosting minimizes loss function,

$$\mathcal{L}_{D^L} = \sum_{\mathbf{x} \in D^L} \mathcal{L}(\mathbf{x}, y) = \sum \mathbf{x} \in D^L e^{-yH(\mathbf{x})} \tag{A.13}$$

To exploit unlabeled data, loss function is modified to take both labeled and unlabeled data into consideration. In SemiBoost [26], unlabeled data is connected with labeled data using similarity measure, $S(\mathbf{x}_i, \mathbf{x}_j)$. All loss function are list as follows.

For labeled samples,

$$\mathcal{L}^L(\mathbf{x}_i, y_i) := e^{-2y_i H(\mathbf{x}_i)}. \tag{A.14}$$

For pair of labeled and unlabeled samples,

$$\mathcal{L}^{LU}(\mathbf{x}_i, y_i, \mathbf{x}_j) := S(\mathbf{x}_i, \mathbf{x}_j) e^{-2y_i H(\mathbf{x}_j)}. \tag{A.15}$$

For pair of two unlabeled samples

$$\mathcal{L}^{UU}(\mathbf{x}_i, \mathbf{x}_j) := S(\mathbf{x}_i, \mathbf{x}_j)\cosh(H(\mathbf{x}_i) - H(\mathbf{x}_j)). \tag{A.16}$$

The combined loss is,

$$\begin{aligned}\mathcal{L} = &\frac{1}{|\chi_L|}\sum_{\mathbf{x}\in\chi_L} e^{-2yH(\mathbf{x})} \\ &+ \frac{1}{|\chi_L||\chi_U|}\sum_{\mathbf{x_i}\in\chi_L}\sum_{\mathbf{x_j}\in\chi_U} S(\mathbf{x}_i, \mathbf{x}_j)e^{-2yH(\mathbf{x})} \\ &+ \frac{1}{|\chi_U||\chi_U|}\sum_{\mathbf{x_i}\in\chi_U}\sum_{\mathbf{x_j}\in\chi_U} S(\mathbf{x}_i, \mathbf{x}_j)e^{-2yH(\mathbf{x})}\end{aligned} \tag{A.17}$$

Now, semi-supervised learning has become an optimization problem by finding the best weak classifier $h_n(\mathbf{x})$ and weight α_n, which is,

$$(\alpha_n, h_n) = \arg\min_{\alpha_n, h_n}(\mathcal{L}). \tag{A.18}$$

Hence, the best $h_n(\mathbf{x})$ is,

$$h_n = \arg\min_{h_n}\left(\frac{1}{|\chi^L|}\sum_{\substack{x\in\chi^L \\ h_n(x)\neq y}}\omega_n(x,y) - \frac{1}{|\chi^U|}\sum_{x\in\chi^U}(p_n(x) - q_n(x))\alpha_n h_n(x)\right), \tag{A.19}$$

where $\omega_n(\mathbf{x}, y)$, $p_n(\mathbf{x})$ and $q_n(\mathbf{x})$ are,

$$\omega_n(\mathbf{x}, y) = e^{-2yH_{n-1}(\mathbf{x})}, \tag{A.20}$$

$$p_n(x) = e^{-2H_{n-1}(x)}\frac{1}{|\chi^L|}\sum_{x_i\in\chi^+} S(x, x_i) + \frac{1}{|\chi^U|}\sum_{x_i\in\chi^U} S(x, x_i)e^{H_{n-1}(x_i) - H_{n-1}(x)}, \tag{A.21}$$

and

$$q_n(x) = e^{2H_{n-1}(x)}\frac{1}{|\chi^L|}\sum_{x_i\in\chi^-} S(x, x_i) + \frac{1}{|\chi^U|}\sum_{x_i\in\chi^U} S(x, x_i)e^{H_{n-1}(x) - H_{n-1}(x_i)}. \tag{A.22}$$

The weight α_n can be got by taking derivative of Eq. A.19, which is,

$$\alpha_n = \frac{1}{4}\ln\left(\frac{\frac{1}{|\chi^U|}\left(\sum_{\substack{x\in\chi^U \\ h_n(x)=1}} p_n(x) + \sum_{\substack{x\in\chi^U \\ h_n(x)=-1}} q_n(x)\right) + \frac{1}{|\chi^L|}\sum_{\substack{x\in\chi^L \\ h_n(x)=y}}\omega_n(x,y)}{\frac{1}{|\chi^U|}\left(\sum_{\substack{x\in\chi^U \\ h_n(x)=1}} q_n(x) + \sum_{\substack{x\in\chi^U \\ h_n(x)=-1}} p_n(x)\right) + \frac{1}{|\chi^L|}\sum_{\substack{x\in\chi^L \\ h_n(x)\neq y}}\omega_n(x,y)}\right). \tag{A.23}$$

Semi-supervised learning makes strong model assumption [25]. Those unlabeled data does not always help [35]. The critical point is in the connection of labeled and unlabeled data. For example, SemiBoost using similarity measure to cross the gap. Grabner et al. [16] then use online version for object tracking. Limitations from tracking results show that, it does diminish drifting problem, however, it also seriously reduce adaptability.

A.4 Multiple Instance Learning

Multiple instance learning (MIL) is also a research topic conducted for decades. The idea of multiple instance learning is originally proposed in 1990 solving handwritten digit recognition [29]. There are many solutions and our illumination focuses on [29] since Babenko et al. [18] chose this one. Viola et al. have proposed two variants to solve the MIL problem, Noisy-OR Boost and ISR Boost. We only describe Noisy-OR because of Babenko's choices.

Different from boosting, examples are not individually labeled. They reside in bags. An example is indexed with two indices, bag index i and instance index j. The probability of an example is positive is,

$$p_{ij} = \frac{1}{1 + e^{-y_{ij}}}. \tag{A.24}$$

Here $y_{ij} = C(x_{ij})$ is the score of the sample and $C(x_{ij}) = \Sigma_t \lambda_t c^t(x_{ij})$ is a weighted sum of weak classifiers. The probability that the bag is positive is,

$$p_i = 1 - \prod_{j \in i} (1 - p_{ij}). \tag{A.25}$$

Under this model, the likelihood for training bags is,

$$L(C) = \prod_i p^{t_i} (1 - p_i)^{(1-t_i)}, \tag{A.26}$$

where $t_i \in \{0, 1\}$ is the categories of bag i. Eq. A.26 could be regarded as objective function. According to AnyBoost approach,

the weight on each example is given as the derivative of the cost function with respect to a change in the score of the example [29].

The derivative of the log likelihood is,

$$\frac{\partial \log L(C)}{\partial y_{ij}} = \omega_{ij} = \frac{t_i - p_i}{p_i} p_{ij}, \tag{A.27}$$

which is the sample weight. Each round, boosting is searching for a classifier which maximizes $\Sigma_{ij} c(x_{ij}) \omega_{ij}$. To better illustrate, equation of step-wise loss function in online MILBoost is adopted as,

$$\mathcal{L}^m = \Sigma_i \left(y_i \log\left(p_i^m\right) + (1 + y_i) \log(1 - p_i^m) \right) \tag{A.28}$$

This is step-wise logistic loss function. Online MILBoost finds the weak classifier with the maximum $\mathcal{L}^m$. Please refer to Eq. A.26.

Observe Eq. A.28. The example weight is consist of bag weight $W_{bag} = \frac{t_i - p_i}{p_i}$ and instance weight $W_{instance} = p_{ij}$. W_{bag} for negative example is always -1. The example weight on positive instance is more complex. When incoming sample is approaching the target, the weight on entire bag is reduced, i.e.,W_{bag} is increasing. In this manner, examples around target is weighted higher than those which is more far away.

In conclusion, we have introduced four different learning methods. First, boosting is an ensemble learning method. Online boosting is its online version. Modifications have made on weight sample adjustment. The primary achievement is the convergence of online boosting to batch boosting. Semi-supervised learning bridge the gap between labeled data and unlabeled data. Finally, multiple instance learning provides the bag concept on target. Every profile from different aspects are one of the target. MIL breaks the sharp boundary between positive samples and negative sample. To sum up, different learning algorithms help object tracking. Although not all of them are suitable in tracking field, we're approaching the ultimate goal of tracking, identifying object identity.