利用隱藏式馬可夫模型之棒球精彩事件短片偵測

研究生：蔡維晉　　指導教授：李素瑛

國立交通大學資訊科學與工程研究所

摘要

近年來棒球影像分析已有相當多的研究成果，但是對於精彩內容意涵事件偵測分類等影像處理技術的分析細膩度尚嫌不足。而這篇論文提出一個有效且有效率的棒球影片精彩內容意涵事件偵測分類系統，利用棒球影片中特定的場地規格及規律的場景轉換，使得此系統能夠辨識正在進行的精彩內容意涵事件。為了達到此一目標，本論文提出的系統概述如下:首先，我們可將一部棒球影片切割成精采短片，而每一個精彩短片開始於投手投球並結束於某些特定結束畫面，利用短片中每張畫面出現的物件特徵及順序，進而利用擷取出來的資訊用來發展以隱藏式馬可夫模型所建立的十二種精彩短片事件分類器。我們在此篇論文提出更精確的特徵擷取，使得棒球精采短片在分類上有著高準確性並且多樣性。此方法在運算上非常有效率，更重要的是，各種精采短片分類的實驗數據結果顯示相當良好的系統效能及準確性。

檢索詞: 精采內容意涵事件偵測分類，特徵或物件擷取，隱藏式馬可夫模型。

Baseball Event Semantic Exploring System Using HMM

Abstract

Despite a lot of research efforts in baseball video processing in recent years, little work has been done in analyzing the detailed semantic baseball event detection. This thesis presents an effective and efficient baseball event classification system for broadcast baseball videos. Utilizing the strictly-defined specifications of the baseball field and the regularity of shot transition, the system recognizes highlight in video clips and identifies what semantic baseball event of the baseball clips is currently proceeding. The semantic exploring system is proposed to achieve the objective. First, a video is segmented into several highlights starting with a PC (Pitcher and Catcher) shot and ending up with a close-up or some specific shots. Before every baseball event classifier is designed, several novel schemes including some specific features such as soil percentage and objects extraction such as first base are applied. The extracted midlevel cues are used to develop baseball event classifiers based on an HMM (Hidden Markov model). Due to specific features detection the proposed method not only improves the accuracy of the highlight classifier but also supports variety types of the baseball events. The proposed approach is very efficient. More importantly, the simulation results show that the classification of twelve significant baseball highlights is very promising.

Index: highlight detection and semantic baseball event classification, features or objects extraction, Hidden Markov Model.

Acknowledgement

I greatly appreciate the kind guidance of my advisor, Porf. Suh-Yin Lee. Without her graceful suggestion and encourage, I cannot complete this thesis.

Besides, I want to give my thanks to my friends and all members in the Information System laboratory for suggestion, especially Mr. Hua-Tsung Chen.

Finally, I would like to express my appreciation to my parents for their supports. This thesis is dedicated to them.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

In recent years, the amount of multimedia information has grown rapidly. This trend leads to the development of efficient sports video analysis. Automatic sports video analysis has attracted considerable attention, because sport video appeals to large audiences. The possible applications of sports video analysis have been found almost in all sports, among which baseball is a quite popular one. However, a whole game is very long but the highlight is only a small portion of the game. In addition, highlight can be detected to provide a tactic for coaching. Based on these motivations, development of the highlight semantic exploring system for the baseball games is our focus.

Because the positions of cameras are fixed in a game and the ways of showing game progressing are similar in different TV channels. Each category of semantic baseball event usually has a similar shot transition. For example, a typical fly out can be composed of a pitch shot followed by an outfield left or center or right shot and then a play in grass shot. Based on this observation, many methods are applied on semantic baseball event detection such as HMM [5][6][7], temporal feature detection [8], BBN (Bayesian Belief Network) [11]. The existing highlight detection and classification systems suffer from at least one of the flaws in the following: (1) only few highlights or mid-level semantics (lower than highlight semantics) are detected, (2) the accuracy of classification is not high enough for practical usage, and (3) time complexity is rather high. To solve the problems of existing highlight detection or classification systems, high accuracy and more specific highlight especially hitting highlight (ball has been hit) detection and classification, is our foremost target.

This thesis presents an HMM-based mechanism to detect and classify baseball events. To improve the accuracy of baseball event classification and specific baseball

event classification, more features and objects (lower than highlight semantics) must be detected. Twelve semantic baseball event types in baseball games are defined and detected in the proposed system: (1) single (2) double (3) pop up (4) fly out (5) ground out (6) two base hit (7) right foul ball (8) left foul ball (9) foul out (10) double play (11) home run (12) home base out. Some mid-level semantics are introduced in the following section and these mid-level semantics are used to detect and classify baseball events. In the proposed framework, highlight detection and baseball event classification in broadcast baseball videos will be more powerful and practical, since comprehensive and detailed information about the game can be presented to users.

The rest of the thesis is organized as the follows. The background and related works are introduced in chapter 2. In chapter 3, we introduce the HMM concept used in our system. Chapter 4 introduces our proposed system including feature extraction, frame classification, baseball event classification. Chapter 5 shows the experimental results and discussion. Finally, conclusion and future work are made in chapter 6.

Chapter 2
Background and Related Works

In chapter 2, the baseball highlight detection in recent years will be introduced. First of all, we will describe the hierarchical structure of a baseball game in section 2-1. In section 2-2, image processing of color space conversion from RGB to HSI is introduced to make some tasks easily such as less influence on luminosity. In the following sections, some related works in PC (Pitcher and Catcher) shot detection, and highlight detection for baseball videos are depicted.

2.1 Hierarchical Structure of Baseball Game

A baseball game is composed of some highlights. Highlight is a sequence of specific shot transition. A shot consists of several similar frames. Different frames have different color distribution, features, and objects (base, line, auditorium, etc.). **Fig. 2-1** shows the concept about hierarchical structure of a baseball game.

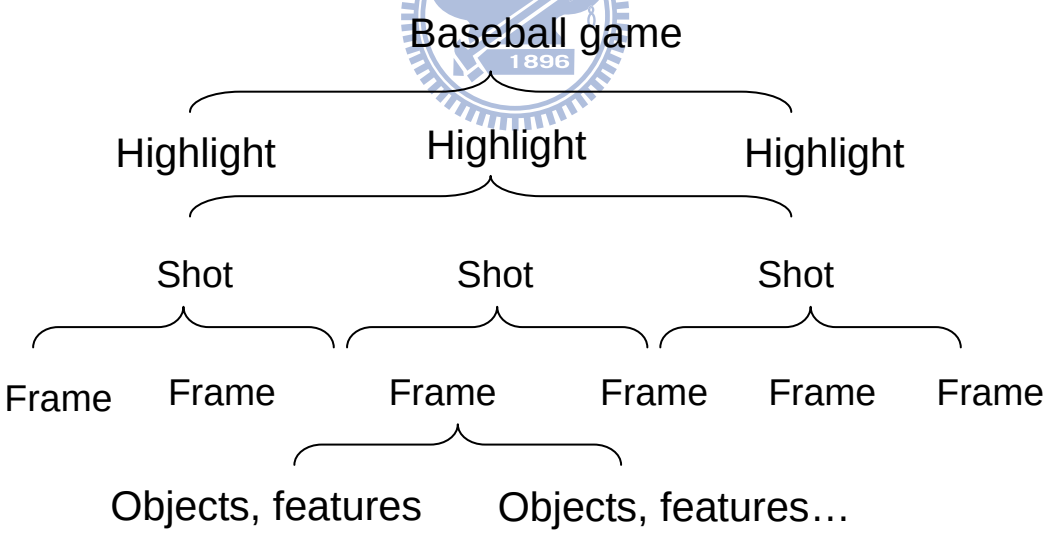


Fig. 2-1 Hierarchical structure of baseball game.

2.2 Color Conversion from RGB to HSI

In image processing, color is an important feature. The influence on luminosity of HSI is less than that of RGB. To make feature extraction or objection detection easily, we can use the following formula as described in **Fig. 2-2** to convert from

RGB to HSI. Similar skill can be found in other sports such as basketball [19].

$$r = \frac{R}{R+G+B} \qquad g = \frac{G}{R+G+B} \qquad b = \frac{B}{R+G+B}$$

$$h = \cos^{-1}\left[\frac{0.5\times[(r-g)\times(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{\frac{1}{2}}}\right] \qquad h\in[0,\pi], b\le g$$

$$h = 2\pi - \cos^{-1}\left[\frac{0.5\times[(r-g)\times(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{\frac{1}{2}}}\right] \qquad h\in[0,\pi], b> g$$

$$s = 1-3\times\min(r,g,b) \qquad s\in[0,1]$$

$$i = \frac{R+G+B}{3\times 255} \qquad i\in[0,1]$$

Fig. 2-2 The conversion from RGB to HSI.

2.3 Pitcher and Catcher Shot (PC shot) Detection

Every baseball highlight starts with PC shot and ends up with some specific shots or a close-up shot, so the PC shot detection plays an important role in baseball highlight detection. The proposed method in [1] based on feature mining can find the effective feature types, the location of the features and threshold values during the learning process.

In general, the composition of the PC shot is auditorium, player, soil and grass. However, it has some unstable elements. The location and uniform of players would be changing. Features without influences on these changes (i.e. location of player) are that the PC shot is composed of ground, wall, and audience. To find the features, an image is divided into 12 × 8 blocks as shown in **Fig. 2-3.**

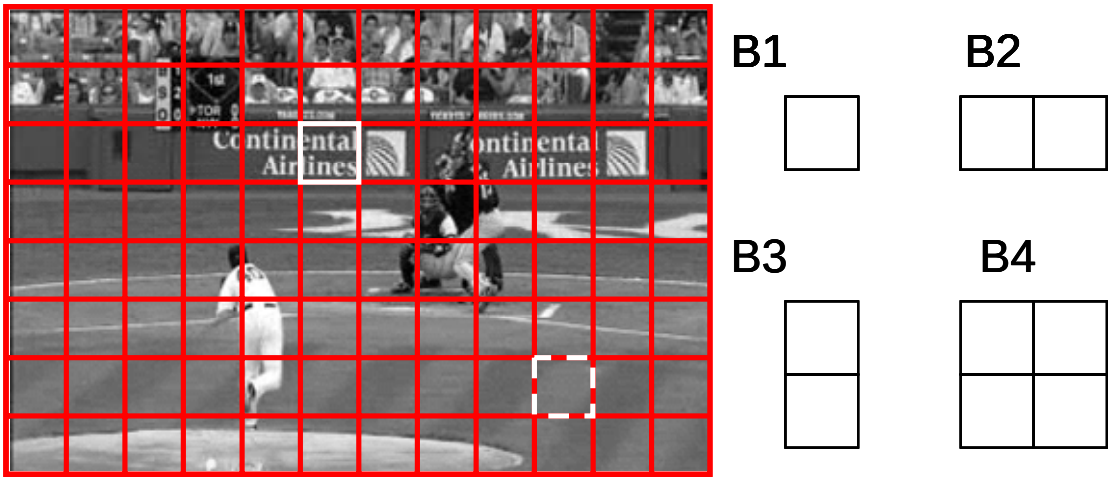


Fig. 2-3 The block types in a frame.

Then we use mean, variance, and log variance on luminosity data to discover the effective blocks (effective block will be elaborated later) for PC shot discrimination in training data. In experiment, we use four block size as a unit, B1, B2, B3, and B4 to calculate the mean, variance, and log variance. Some trends are observed as follows.

(1) The mean of the luminosity in the ground block, as shown by white dotted line in **Fig. 2-3,** is stable even if a camera shift takes place.

(2) The variance of the luminosity in the ground block is small because the ground is flat in the block.

(3) In a wall block as shown by white solid line, the variance becomes large due to the high texture, but the log variance can be assumed to be stable in the block.

We assume that fusion of these three features is effective in PC shot discrimination. Next, desirable features should be stable at the same location in the training data set, so the block with small variance of the feature called effective block in the training data is thought of as the best location of those blocks.

The X-axis of an image is divided into 12 blocks and the Y-axis is divided into 8 blocks of the image as shown in Fig. 2-3. Four block types B1, B2, B3 and B4 are used to search for the effective blocks for PC shot discrimination. The mean $M_{f,t,z}$, the variance $V_{f,t,z}$, and the log variance $LV_{f,t,z}$ computed within the block type t and the

location i B_i^t, is defined by Eq.(l), Eq.(2), and Eq.(3). Here, t (t = 1~4) and i (i = 1~96, 1~48, or 1~24 depends on different block type) indicate the block type and block position counted from top-left corner in a frame f respectively. Let Gray(x, y) denote the luminosity at location (x, y) of the image and |B| denote the number of pixels in a block.

$$M_{f,t,i} = \frac{1}{\left|B_i^t\right|} \sum_{x,\, y \in B_i^t} Gray\ (x, y) \qquad (1)$$

$$V_{f,t,i} = \frac{1}{\left|B_i^t\right|} \sum_{x,\, y \in B_i^t} \left(Gray\ (x, y) - M_{f,t,i}\right)^2 \qquad (2)$$

$$LV_{f,t,i} = \frac{1}{\left|B_i^t\right|} \sum_{x,\, y \in B_i^t} \log\left(Gray\ (x, y) - M_{f,t,i}\right)^2 \qquad (3)$$

Let N be the number of PC shots in training data. The variance of the mean in the block B_i^t among the training data is defined in Eq. (4).

$$V_{M,t,i} = \frac{1}{N} \sum_{f=1}^{N} \left(M_{f,t,i} - \frac{1}{N} \sum_{f=1}^{N} M_{f,t,i} \right)^2 \qquad (4)$$

When all $V_{M,t,i}$ for all t and i are placed in ascending order, the variance at rank *n* is defined as V^n_{M,t_M,i_M}. The block $B_{i_M}^{t_M}$ at the rank 1 with variance V^1_{M,t_M,i_M} is regarded as the optimal block in mean luminosity. Max and min value (threshold) of block luminance mean M_{f,t_M,i_M} at frame f in block $B_{i_M}^{t_M}$ is defined by Eq. (5) and (6) respectively.

$$M^{max}_{t_M,,i_M} = \max_f \left(M_{f,t_M,i_M}\right) \qquad (5)$$

$$M^{min}_{t_M,,i_M} = \min_f \left(M_{f,t_M,i_M}\right) \qquad (6)$$

Similarly, variance of variance $V_{V,t,i}$ and variance of log variance $V_{LV,t,i}$ are

calculated respectively. The optimal block in variance and log variance are found of rank 1 $V_{V,t,i}^{1}, V_{LV,t,i}^{1}$. Max and min value (threshold) of block luminance variance and log variance are $V_{t_M,,i_M}^{\max}, V_{t_M,,i_M}^{\min}, LV_{t_M,,i_M}^{\max}, LV_{t_M,,i_M}^{\min}$. Last, if a test image f' meets the conditions in Eq. (7), the image f' is viewed as a PC shot.

$$\begin{aligned} M_{t_M,,i_M}^{\min} &\le M_{f',t_M,,i_M} \le M_{t_M,,i_M}^{\max} \\ V_{t_M,,i_M}^{\min} &\le V_{f',t_M,,i_M} \le V_{t_M,,i_M}^{\max} \\ LV_{t_M,,i_M}^{\min} &\le LV_{f',t_M,,i_M} \le LV_{t_M,,i_M}^{\max} \end{aligned} \quad (7)$$

This method [1] showed 95.5% accuracy at F-measure score within 1/30 of real time.

2.4 Highlight Detection and Classification

Highlight detection and classification is a popular issue as a result of the following reasons: (1) Provide coach with a guidance of tactic, (2) Make a highlight movie, (3) Index each highlight used for baseball event retrieval, and (4) More accurate in baseball event classification from baseball game. In the past few years, significant research [5, 6, 7, 8, 9, 10, 11] has been devoted to the content analysis of baseball game. [5][6][7] use the statistical model of HMM to detect and classify the highlights. For example, Chang et al. [5] assumes that most highlights in baseball games consist of certain shot types and these shots have similar transition in time. Each highlight is described by an HMM as shown in **Fig. 2-4** and each hidden state is represented by its predefined shot types as shown in **Fig. 2-5**. Some features are used as observations to train the HMM model for highlight recognition. [5][6] use some features and shots as observations and states in HMM for highlight classification. Low accuracy and few highlight types are the main disadvantages because the information is too little to detect various highlights and to get high accuracy. [8] records some objects or features such as field type, speech, and camera motion start

time and end time to find the frequent temporal patterns as shown in **Fig. 2-6** for highlight detection and classification. The accuracy in [8] is better than that in [5][6], but they use speech, caption and shot as features so that the cost of time complexity is high. [9][11] combine some shots such as pitch and catch, infield, outfield, and non-field shot with scoreboard as shown in **Fig. 2-7** and **Fig. 2-8** as medium-level cues, and then use Bayesian Belief Network (BBN) structure for highlight classification. [10] uses some condition rules for highlight classification. [9][11] use scoreboard as additional information so that the accuracy is very high, but the rough shot classification lead to the low variety of hitting highlight.

In this thesis, we will emphasize the variety of baseball events and high accuracy via more features and objects exploration. Statistical model HMM is used for highlight classification and HMM concept will be elaborated in chapter 3.

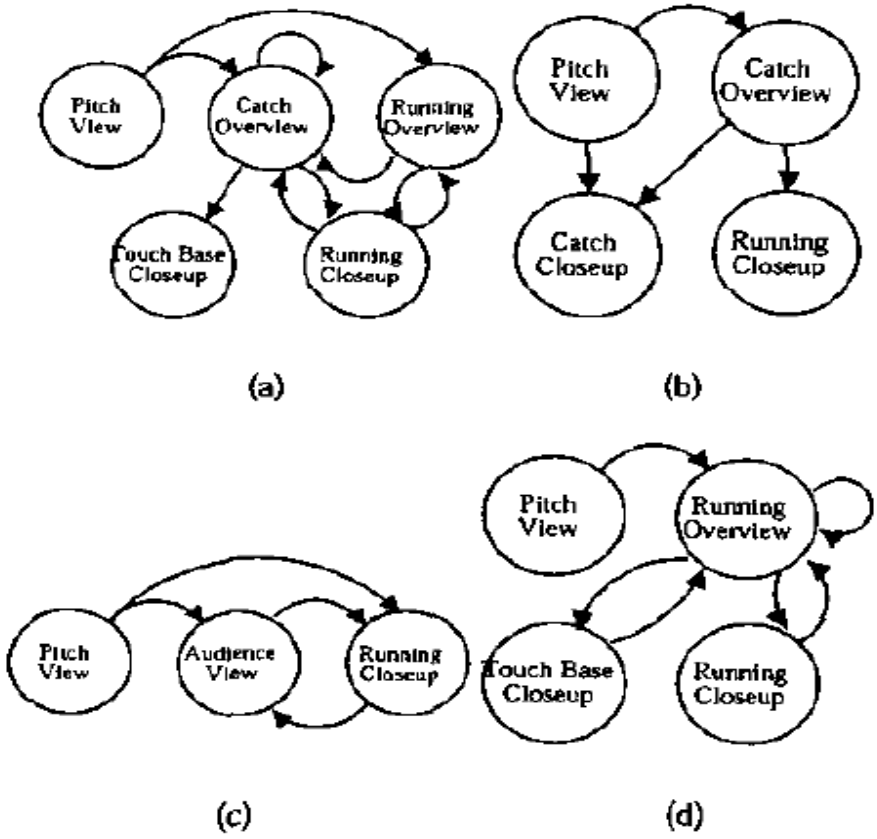


Fig. 2-4 Four baseball HMMs defined in [5] (a) nice hit, (b) nice catch, (c) homerun, and (d) the play within the diamond (events occur in infield)

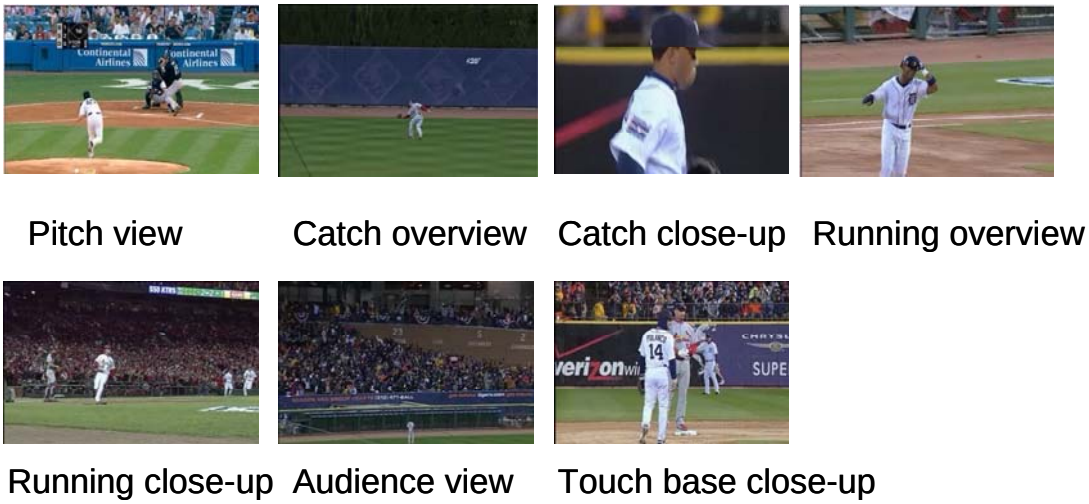


Fig. 2-5 The seven pre-defined types of shots in [5].

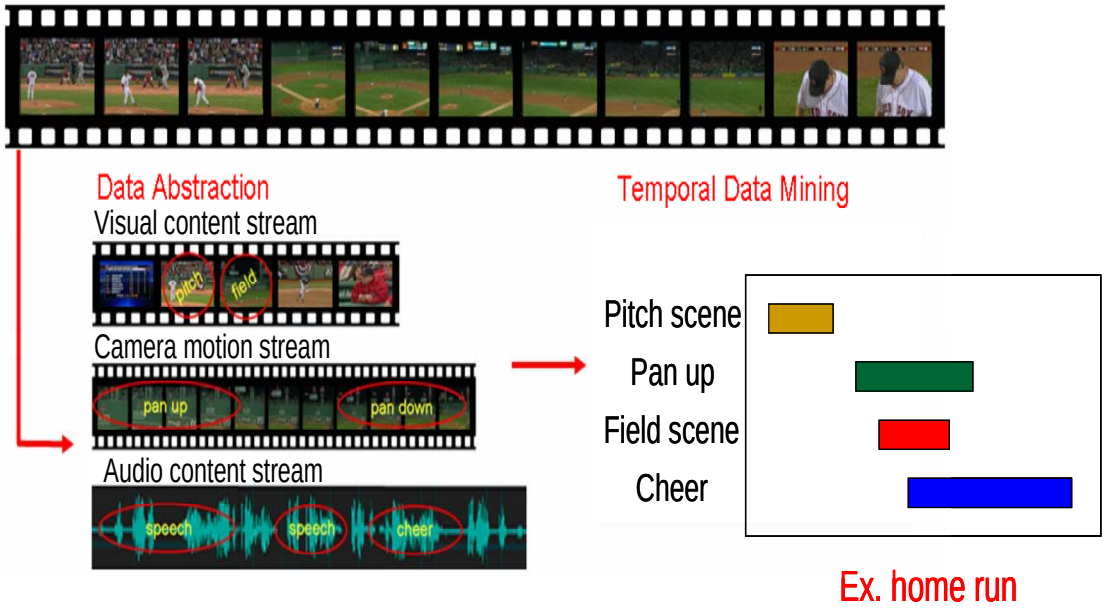


Fig. 2-6 The system overview of highlight detection and classification in [8].

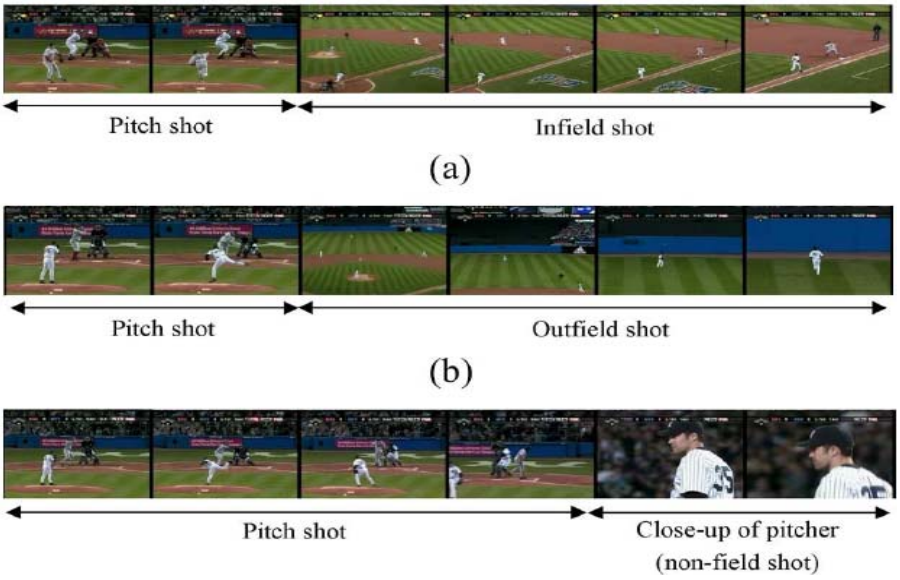


Fig. 2-7 Three shot transition types defined in [11].

Fig. 2-8 Twelve scoreboards defined in [11].

Chapter 3
Hidden Markov Model

Real-world processes generally produce observable outputs which can be characterized as signals. Finding the regular rule in those signals is a popular issue in real world (e.g., the sequence of instruction in computer, the sequence of speech recognition, etc). It is required to build signal models to analyze the real-world signal, and then these signal models can be realized into applications in practical systems such as prediction system, recognition system, identification system, etc.

Generally, signal models can be classified into deterministic models, and statistical models. In practical systems, deterministic models are used to exploit the specific rules of the signal such as traffic light. The current state can be determined easily by the previous state. The other case of signal model is statistical model which is modeling a wide range of time series data like Poisson process, Markov model, Hidden Markov Model and so on. In this case, the next state cannot be determined by current state, but a model is still created to estimate signal properties even if the model would miss some messages.

Among statistical models, Hidden Markov model is a powerful statistical model for modeling the generative sequence in many fields such as biology, mathematics, speech recognition, signal processing. Differing from Markov model, the state is not observable. Because features and shot transitions can be viewed as observable outputs and states, a highlight can be described by a signal model such as HMM. When an observable signal is given, the likelihood was computed by each signal model and the best match is the proposed highlight.

3.1 Element of an HMM

A Hidden Markov model has several states, each of which has a transition probability from current state to next state and the next state is only dependent on the

current state. Each state has several output symbols but yields a symbol at one time. Each symbol has an output probability and the output symbol at time t is dependent only on the current state. Some notations are defined as follows:

T = length of the observation sequence.

N = the number of states in the model.

M = the number of observation symbols.

The set of N states: $Q = \{q_1, q_2, ..., q_N\}$

The set of M output symbols (observations): $V = \{v_1, v_2, ..., v_M\}$

State s_t : Which state belongs to at time t, $s_t \in Q$ (unobservable). For example,

$s_t = q_i$ is a representation of staying in state q_i *at time t*.

The state transition probability: $\boldsymbol{A} = \{a_{ij} \mid a_{ij} = P_r(s_{t+1} = q_j \mid s_t = q_i) \mid 1 \le i, j \le N\}$

The output symbol probability: $\boldsymbol{B} = \{b_j(k) = P_r(v_k \mid s_t = q_j) \mid 1 \le j \le N, 1 \le k \le M\}$

The initial probability: $\boldsymbol{\pi} = \{\pi_i \mid \pi_i = P_r(s_1 = q_i) \mid 1 \le i \le N\}$

Parameter set of HMM model $\boldsymbol{\lambda} = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}\}$

Observed symbol sequence $\boldsymbol{O} = o_1, o_2, ..., o_T \, (length = T)$

A Hidden Markov model can be described by a given tuple λ as shown in **Fig. 3-1.**

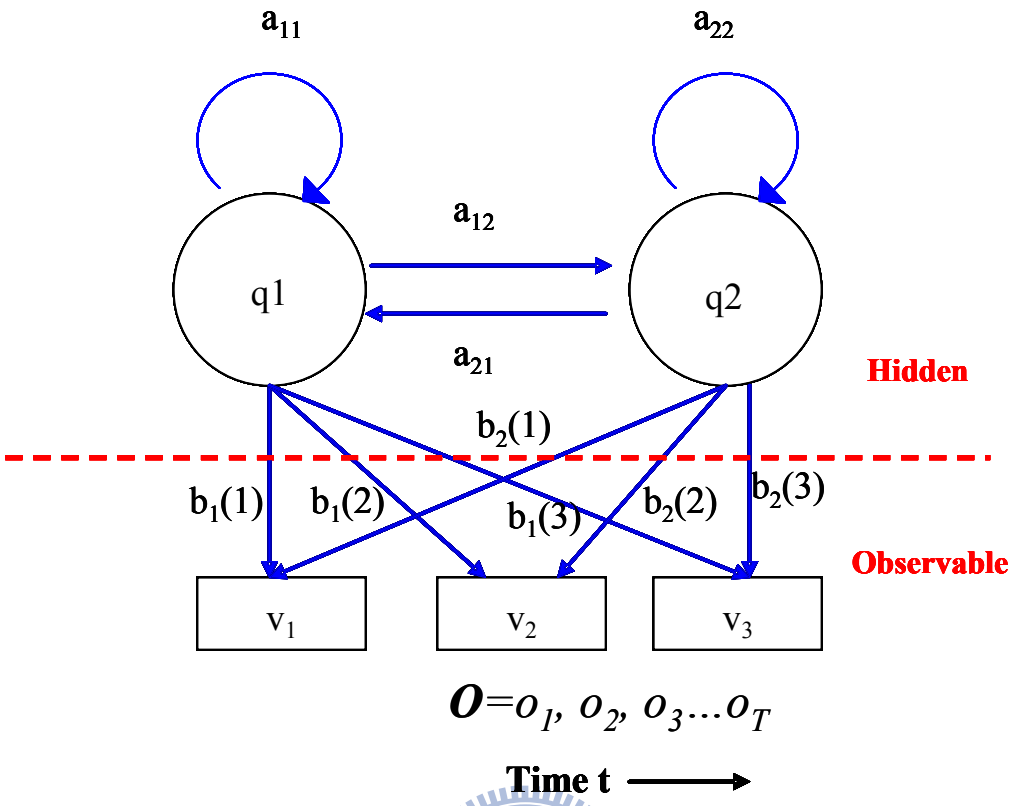


Fig. 3-1 Illustration of the concept of HMM

Figure 3-1 illustrates the HMM concept. Two states q_1, q_2, and three output symbols v_1, v_2, and v_3 are shown. The transition probability from q_i to q_j is marked as directed line a_{ij}. Each state could produce an observation at one time and each observation is assigned an output probability.

The first time to enter which state of HMM is stochastically determined by an initial state matrix $\boldsymbol{\pi}$. The transition probability of each state to other state is determined by a transition probability matrix **A**. If there are N states, the matrix **A** is an N×N matrix. Note that the HMM can transit from a state to itself. Each state of the HMM stochastically outputs a symbol at a time determined by a matrix B. If there are M output symbols, the matrix **B** is an N×M matrix. Time from 1 to T, the HMM will output symbol $\boldsymbol{O} = o_1, o_2, ..., o_T$, but the state transition sequence is non-observable.

A recognition process can be proceeded by given the tuple λ = **{A, B, $\boldsymbol{\pi}$}** of an HMM model, and the matrix **A**, **B**, and $\boldsymbol{\pi}$ can be learned in HMM training stage.

In recognition process, the HMM with the highest probability will be chosen as a recognized result. Recognizing time-sequential symbols is equivalent to determining which HMM produce the output symbols. Section 3.2 and section 3.3 will describe the recognition process and the learning process for an HMM signal model.

3.2 Recognition Process in HMM

One HMM is created for each category for recognizing time-sequential observed symbols. In recognition phase, we will compute the probability $P_r(\boldsymbol{O}|\ \lambda)$ for each category and the best matches will be chosen as the proposed answer from all HMMs of a given observation $\boldsymbol{O} = o_1, o_2, ..., o_T$. That is,

Give an observation, $\boldsymbol{O} = o_1, o_2, ..., o_T$

Each HMM has a tuple $\lambda_i = \left(\mathbf{A}^i, \mathbf{B}^i, \boldsymbol{\pi}^i\right)\ \ i = 1,2,...,C$ (if there are C categories)

Proposed answer = arg max ($P_r(\lambda_i|\ \boldsymbol{O})$)

The probability of the observation sequence of a given model is equivalent to evaluating how well a model predicts a given observation sequence. So now the current problem is how to compute the probability $P_r(\boldsymbol{O}|\ \lambda)$ of an observation sequence $\boldsymbol{O}$ and a given HMM λ.

The most straightforward way to compute the probability of the observations $\boldsymbol{O}$ ($\boldsymbol{O} = o_1, o_2 \ldots o_T$) for a specific state sequence Q ($Q=q_1q_2q_3....q_T$) is:

$$P(\boldsymbol{O} \mid Q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) = b_{q1}(o_1) \times b_{q2}(o_2) \times ... \times b_{qT}(o_T) \qquad (8)$$

and the probability of the state sequence is:

$$P(Q \mid \lambda) = \pi_{q1} a_{q1q2} a_{q2q3} a_{q3q4} ... a_{qT-1qT} \qquad (9)$$

So we can calculate the probability of the observations given the model as:

$$P(\boldsymbol{O}|\lambda) = \sum_{Q} P(\boldsymbol{O}|Q,\lambda) P(Q|\lambda) = \sum_{q1q2..qT \in all\ path} \pi_{q1} b_{q1}(o_1) a_{q1q2} b_{q2}(o_2) a_{q2q3} .. a_{qT-1qT} b_{qT}(o_T) \qquad (10)$$

The problem $P_r(\boldsymbol{O}|\ \lambda)$ can be solved by the Eq. (10), but the number of operations

involved in the calculation is in the order of N^T. This is very time consuming even if the length of the sequence T is moderate.

It is observed that many redundant calculations would be made by directly evaluating Eq. (10), and therefore caching the intermediate calculations can result in time complexity reduction. The cache is implemented as a trellis of states at each time stage, calculating the cached value (called α) for each state as a sum over all states at the previous time step. α is the probability of the partial observation sequence $o_1\ o_2\ \ldots\ o_t$ in state q_i at time t. The concept is shown in **Fig. 3-2** and the forward probability variable is defined in Eq. (11).

$$\alpha_t(i) = P_r(o_1, o_2, \ldots, o_t, s_t = q_i \mid \lambda) \qquad (11)$$

Eq. (11) describes the probability of the partial observation sequence from 1 to t, ending in state q_i.

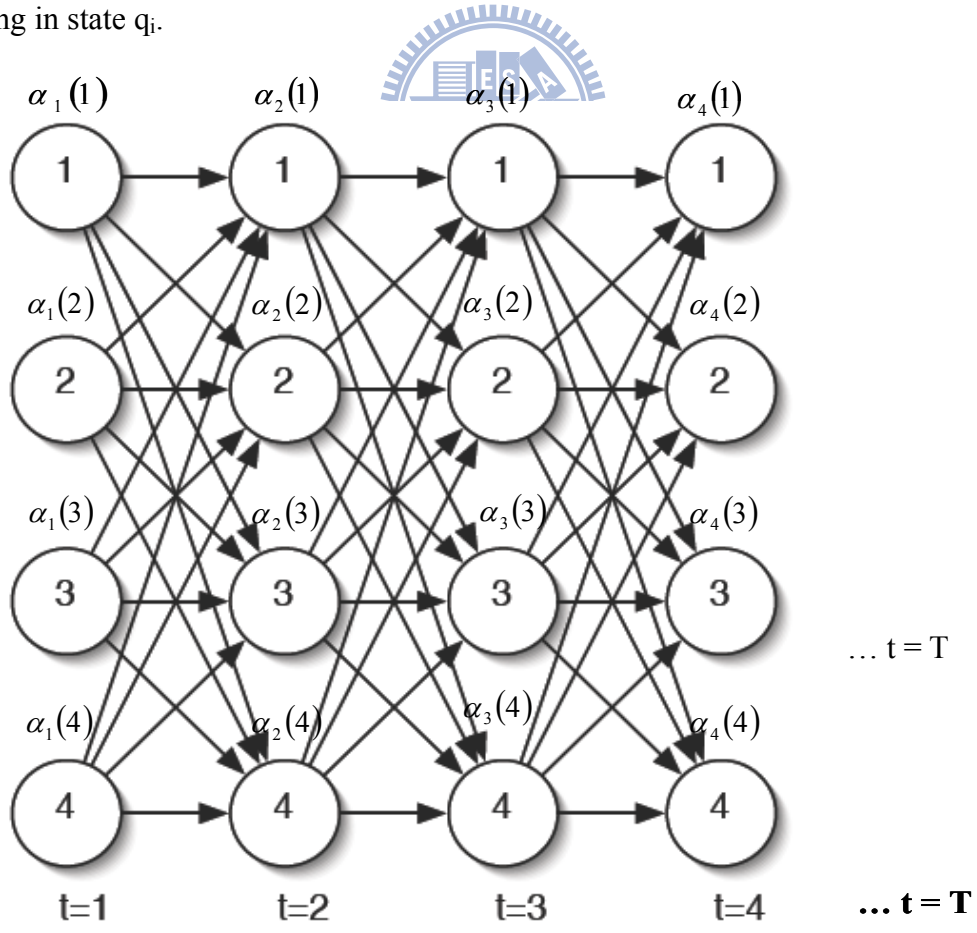


Fig. 3-2 Illustration of the forward algorithm of variable $\alpha_t(i)$

In **Fig. 3-2**, in each time step, the partial probability α_t of each state (trellis) is filled and the sum of the final column of the trellis will equal the probability of the observation sequence. The algorithm for this process is called the forward algorithm and is as follows:

1. Initialization

$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N \tag{12}$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}) \quad 1 \le t < T, \quad 1 \le j \le N \tag{13}$$

3. Termination

$$P(\boldsymbol{O} \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{14}$$

The induction step is the key to the forward algorithm as shown in **Fig. 3-3**. In Eq(13), index j and i represent the current state index and previous state index respectively. For each state q_j, $\alpha_t(j)$ stores the probability of arriving in that state having observed the observation sequence up to time T. In termination step, adding up each forward partial variable σ_T is the probability of the observation produced from the HMM model. It is obvious that by caching α values the forward algorithm reduces the time complexity of calculations involved from $2TN^T$ to N^2T.

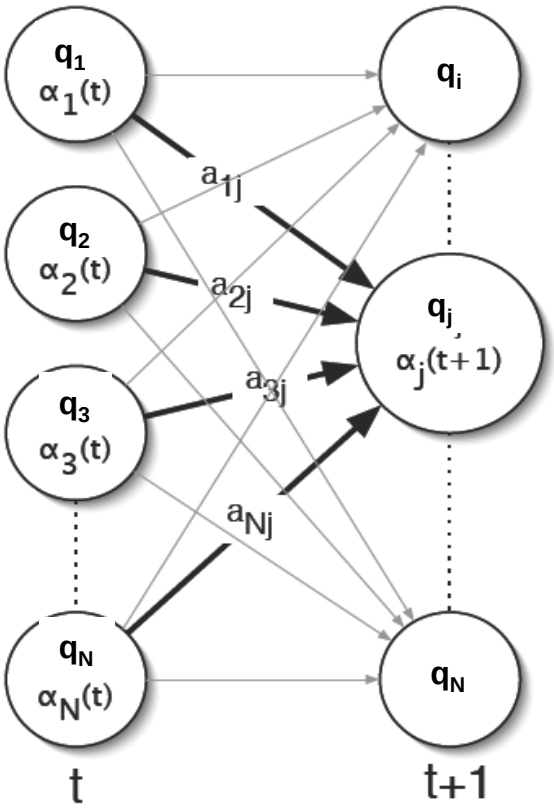


Fig 3-3 The induction step of the forward algorithm

Recognition Process also can be fast solved by backward probability variable similar to forward probability variable α_t and the backward probability variable is defined in Eq. (15).

$$\beta_t(i) = P(o_{t+1}o_{t+2}...o_T \mid s_t = q_i, \lambda) \qquad (15)$$

Eq. (15) describes the probability of the partial observation sequence from time t + 1 to T, starting in state q_i. The algorithm for this process is called the backward algorithm and is as follows:

1. Initialization

$$\beta_T(i) = 1 \qquad 1 \le i \le N \qquad (16)$$

2. Induction

$$\beta_t(j) = \sum_{j=1}^{N} a_{ij}\beta_{t+1}(j)\, b_j(o_{t+1}), \quad 1 \le t < T, \quad 1 \le i \le N \qquad (17)$$

3. Termination

$$P(\boldsymbol{O} \mid \lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1)\beta_1(i) \qquad (18)$$

The initialization step defines $\beta_T(i)$ to be 1 for all state i at time T. The induction step computes the partial probability of all states at time t from time t+1 as shown in **Fig. 3-4**. All possible states q_i at time t+1 account for the transition from q_i to q_j (the a_{ij} term), as well as the observation o_{t+1} in state j (the $b_j(o_{t+1})$ term), and then account for the remaining partial observation sequence from state j (the $\beta_{t+1}(i)$ term). In termination step, calculate $\sum_{i=1}^{N} \pi_i b_i(o_1)\beta_1(i)$ for all state i at time step 1.

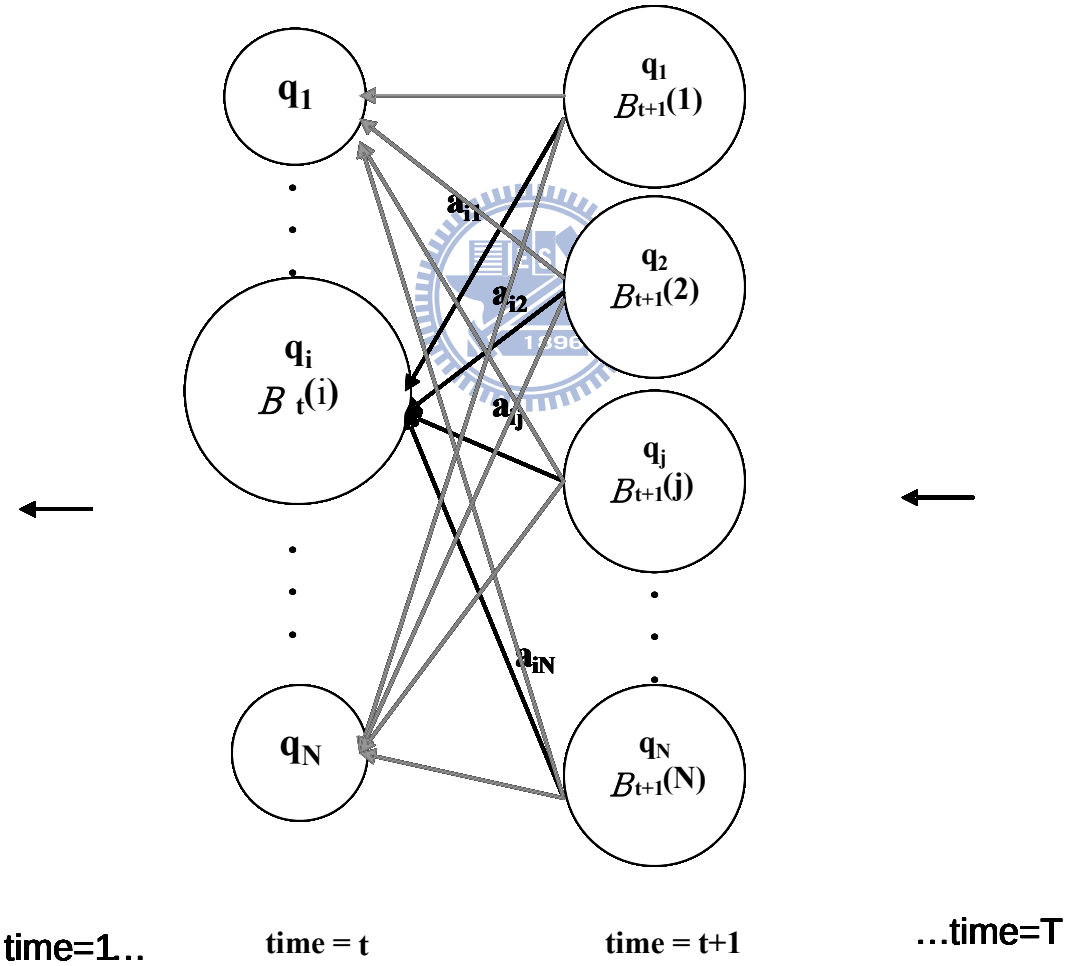


Fig. 3-4 The induction step of the backward algorithm.

We can solve the recognition process problem in time complexity N^2T by using forward algorithm or backward algorithm. Each HMM will output a probability and

the best match will be chosen as a recognized result.

3.3 HMM Training (learning)

The most difficult problem of HMMs is to determine a method to adjust the model parameters λ = **(*A, B, π*)** to maximize the probability of the observation sequence given the model. Given any finite observation sequence as training data, there is no optimal method to estimate the model parameter. However, we can use an iterative procedure such as Segmental K-means algorithm [13] or Baum-Welch algorithm [18] to maximize $P(\boldsymbol{O}, I \mid \lambda)$ (I is the optimal state sequence) or $P(\boldsymbol{O} \mid \lambda)$. In Segmental K-means algorithm the parameters of the model λ = **(A, B, π)** are adjusted to maximize $P(\boldsymbol{O}, I \mid \lambda)$ where I here is the optimal state sequence as given by the Viterbi algorithm [14]. In Baum-Welch re-estimation, here parameter of the model λ = **(A, B, π)** are adjusted so as to increase $P(\boldsymbol{O} \mid \lambda)$ until a maximum value is reached. As seen before, calculating $P(\boldsymbol{O} \mid \lambda)$ involves summing up $P(\boldsymbol{O}, Q \mid \lambda)$ over all possible state sequence $Q(Q = q_1 q_2 q_3 ... q_T)$. Hence Baum-Welch algorithm dose not focus on a particular state sequence. The two methods will be described as follows respectively.

K-means algorithm takes us from λ^k to λ^{k+1} (iteration k to k+1) such that $P(\boldsymbol{O}, I_k^* \mid \lambda^k) \le P(\boldsymbol{O}, I_{k+1}^* \mid \lambda^{k+1})$ where, I_k^* is the optimum state sequence for $\boldsymbol{O} = o_1, o_2 ..., o_T$ and λ^k, found according to the Viterbi algorithm. The criterion of optimization is called the **maximum state optimized likelihood criterion**. This function $P(\boldsymbol{O}, I^* \mid \lambda) = \max_I P(\boldsymbol{O}, I \mid \lambda)$ is called the **state optimized likelihood function**. Training the model in K-means Algorithm, a number of (training) observation sequences are required. Let there be *w* sequences available. Each sequence consists of T observation and each observation symbol (o_i) is assumed to be a vector of dimension $D(D \ge 1)$. K-means Algorithm then consists of the following

steps:

1. Randomly choose N observation symbols (map vector of dimension *D* to symbol by rule table) and assign each of the *w*T observation symbols to one of these N symbols from which its Euclidean distance is minimal. Hence we have formed N clusters, each of which is called a state (1 to N). We can divide those training data into N groups and pick one observation vector from each group. Of course this method is just to make the initial choice of states as widely distributed as possible.
2. Calculate the initial probabilities and the transition probabilities. i, and j represent the current state index and next state index and t represents time from 1 to T-1:

$$\pi_i = \frac{\text{Number of occurrences of } \left\{ o_1 \in state \quad i \right\}}{\text{Total number of sequence}}, \quad 1 \le i \le N \tag{19}$$

$$a_{ij} = \frac{\text{Number of occurrences of } \left\{ o_t \in state \quad i, o_{t+1} \in state \quad j \right\}}{\text{Total number occurrences of } \left\{ o_t \in state \quad i \right\}}$$

$$1 \le i \le N, 1 \le j \le N \tag{20}$$

3. Calculate the mean vector and the covariance matrix for each state: for $1 \le i \le N$ i, and j represents current and next state index, t represents time from 1 to T:

$$\overline{\mu_i} = \frac{1}{N} \sum_{o_t \in state \quad i} o_t \tag{21}$$

$$\overline{V_i} = \frac{1}{N} \sum_{o_t \in state \quad i} \left(o_t - \overline{\mu_i} \right)^T \left(o_t - \overline{\mu_i} \right) \tag{22}$$

4. Calculate the symbol probability distributions for each training vector for each state as (assume Gaussian distribution – change the formulas below for the particular probability distribution that suits problem). For $1 \le i \le N$, i represents state index and t represents time from 1 to T

$$b_i(o_t) = \frac{1}{(2\pi)^{D/2}\left|\overline{V}_i\right|^{1/2}} \exp\left[-\frac{1}{2}(o_t - \overline{\mu}_i)\overline{V}_i^{-1}(o_t - \overline{\mu}_i)^T\right] \quad (23)$$

5. Find the optimal state sequence I^* (as given by Viterbi algorithm) for each training sequence using $\overline{\lambda} = (\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{\pi}})$ computed in step2 to 4 above. $\overline{\boldsymbol{A}}$, $\overline{\boldsymbol{B}}$, *and* $\overline{\boldsymbol{\pi}}$ are the new state transition, output symbol, and initial state probability respectively from re-estimation. Each observation symbol is reassigned a state if its original assignment is different from the corresponding estimated optimum state.
6. If any observation symbol is reassigned a new state in step5, use the new assignment and repeat step2 through step6; otherwise, stop.

It can be shown in [15] that Segmental K-means algorithm converges to the state-optimized likelihood function for a wide range of observation density functions including Gaussian density function.

The second method is called Baum-Welch algorithm, assuming that an initial model can be improved upon by using the Eq. (30)-(32). An initial HMM can be constructed in any way such as random generation, but we may use the first five steps of the Segmental K-means algorithm described above to give us a reasonable initial estimate of the HMM and use Baum-Welch algorithm to re-estimate. Before we get down to the actual Eq. (30)-(32) of Baum-Welch algorithm, some concepts and notations should be introduced that shall be required in the final Eq. (30)-(32).

The forward-backward variable γ_t is defined in Eq. (24).

$$\gamma_t(i) = P(s_t = q_i \mid o_1, o_2, \ldots o_T, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(\boldsymbol{O} \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \quad (24)$$

Eq. (24) describes the probability of being at state q_i in time t. To describe the procedure of re-estimation (iterative update and improvement) of HMM parameter,

the variable $\varepsilon_t(i,j)$ was defined in Eq. (25) and Eq. (25) describes the probability of being at state q_i in time t and at state q_j in time t+1.

$$\varepsilon_t(i,j) = P(s_t = q_i, s_{t+1} = q_j \mid \boldsymbol{O}, \lambda) \tag{25}$$

The sequence of events leading to the conditions required by Eq. (25) is illustrated in **Fig. 3-5**. It should be clear, from the definitions of the forward variable $\alpha_t(i)$ and backward variable $\beta t(i)$, that we can re-write Eq. (25) in the following form

$$\varepsilon_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{P(\boldsymbol{O}\mid\lambda)} = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \tag{26}$$

where the numerator term is just $P(s_t = q_i, s_{t+1} = q_j \mid \boldsymbol{O}, \lambda)$ and the division by $P(\boldsymbol{O}\mid\lambda)$ gives the desired probability measure.

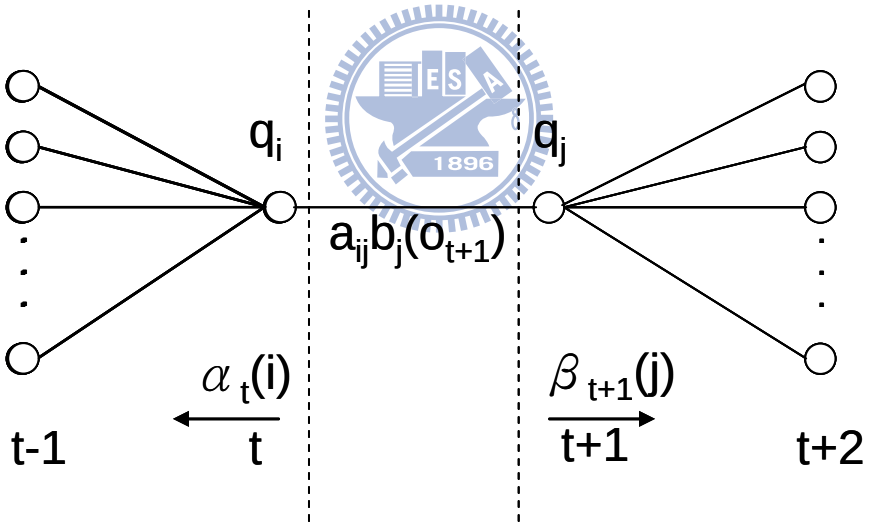


Fig. 3-5 Illustration of the sequence of operations required for the computation of the joint event that the system is in state q_i at time t and q_j at time t+1

If we sum up forward-backward variable $\gamma_t(i)$ from t=1 to T at each state i, we get a quantity which can be viewed as the expected number of times that state q_i is visited, or if we sum up only to T-1 then we shall get the expected number of transitions out of state q_i (as no transition is made at t = T). Similarly if $\varepsilon_t(i,j)$ be summed up from t=1 to T-1, we shall get the expected number of transitions from state q_i to state q_j. Hence

$\sum_{t=1}^{T}\gamma_t(i)$ = expected number of times state q_i is visited. (27)

$\sum_{t=1}^{T-1}\gamma_t(i)$ = expected number of transition from q_i (28)

$\sum_{t=1}^{T-1}\varepsilon_t(i,j)$ = expected number of transitions from q_i to q_j. (29)

Using above formulas we can give a method for re-estimation of the parameter of an HMM. A set of reasonable re-estimation formulas for $\bar{A}$, $\bar{B}$, and $\bar{\pi}$ are

($\bar{A}$ *is new state transition probability matrix*, $\bar{B}$ *is new output symbol distribution matrix, and* $\bar{\pi}$ *is new initial state probability matrix*)

$\bar{\pi}_i$ = expected number of times being in state q_i at time(t = 1) = γ_1(i). (30)

(index i represents the state i)

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } q_i \text{ to state } q_j}{\text{expected number of transitions from state } q_i} = \frac{\sum_{t=1}^{T-1}\varepsilon_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \quad (31)$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } q_j \text{ and observing symbol } v_k\ (k=1\sim M)}{\text{expected number of times in state } q_j} = \frac{\sum_{t=1}^{T}\gamma_t(i)}{\sum_{t=1}^{T}\gamma_t(i)}\Bigg|_{\text{s.t.} o_t = v_k} \quad (32)$$

If the initial model λ = **(A, B, π)** is defined, we use Eq. (27)-(29) to compute the right hand sides of Eq. (30)-(32) for $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$. Baum and his colleagues prove that either (1) the initial model λ defined a critical point of the likelihood function, in which case $\bar{\lambda}$ =λ; or (2) model $\bar{\lambda}$ is more likely than model λ in the sense that $P(O \mid \bar{\lambda}) > P(O \mid \lambda)$, i.e. a new model $\bar{\lambda}$ has been found from which the

observation sequence is more likely to have been produced. We can improve the probability of $\boldsymbol{O}$ which is being observed from the model if we repeat the above procedure and use $\bar{\lambda}$ to replace the λ several times until some limiting point is reached. The final result of re-estimation is called a maximum likelihood estimate of the HMM.

Chapter 4
Proposed scheme for Event classification

4.1 Overview of Proposed Scheme

Overview of the proposed semantic baseball event classification is depicted in **Fig. 4-1(a),** and **Fig. 4-1(b)**. The process can roughly be divided into two steps: training step and classification step. In training step, each type as listed in **Table 4-2** of indexed baseball event was input as training data for each highlight classifier. In classification step, when each observation symbol sequence of unknown clip was input, each highlight classifier will evaluate how well a model predicts a given observation sequence.

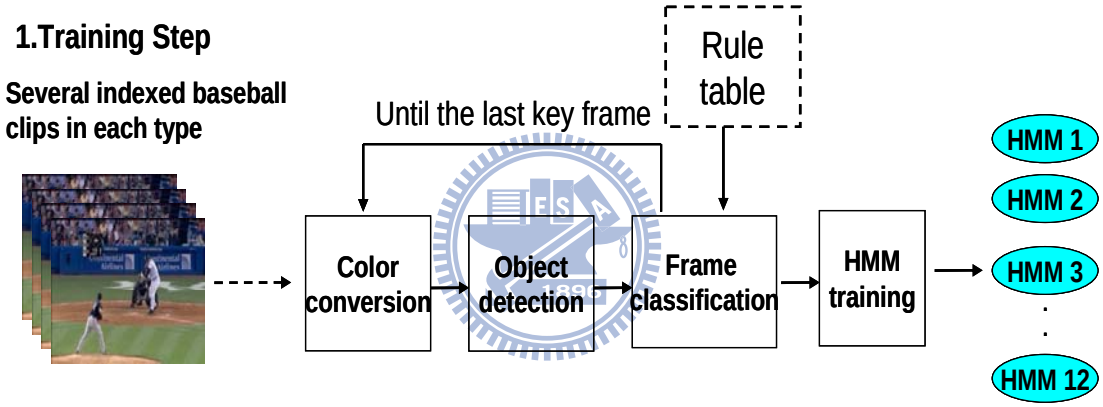


Fig. 4-1(a) Overview of the training step in proposed baseball event classification

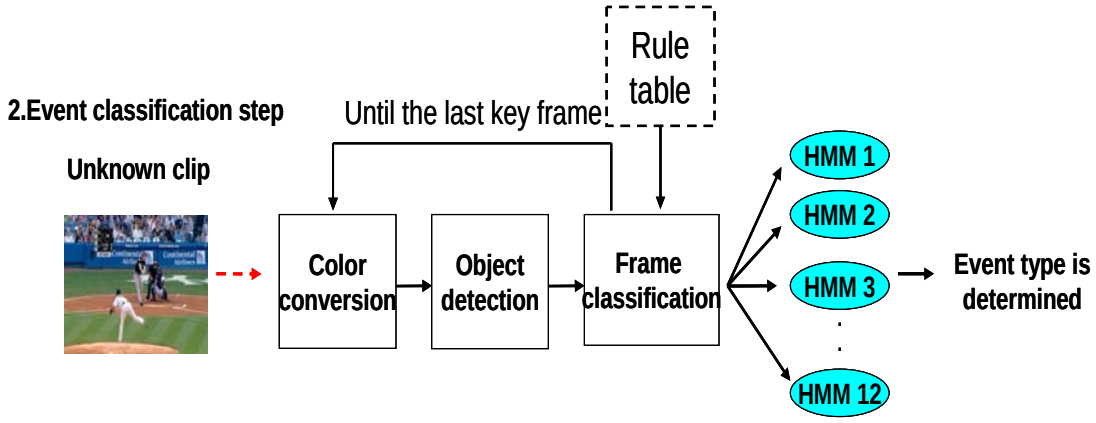


Fig. 4-1(b) Overview of the classification step in proposed baseball event classification

Each highlight clip as input starts with a PC shot and ends up with a close-up shot or a specific shot depending on different baseball event type. There are

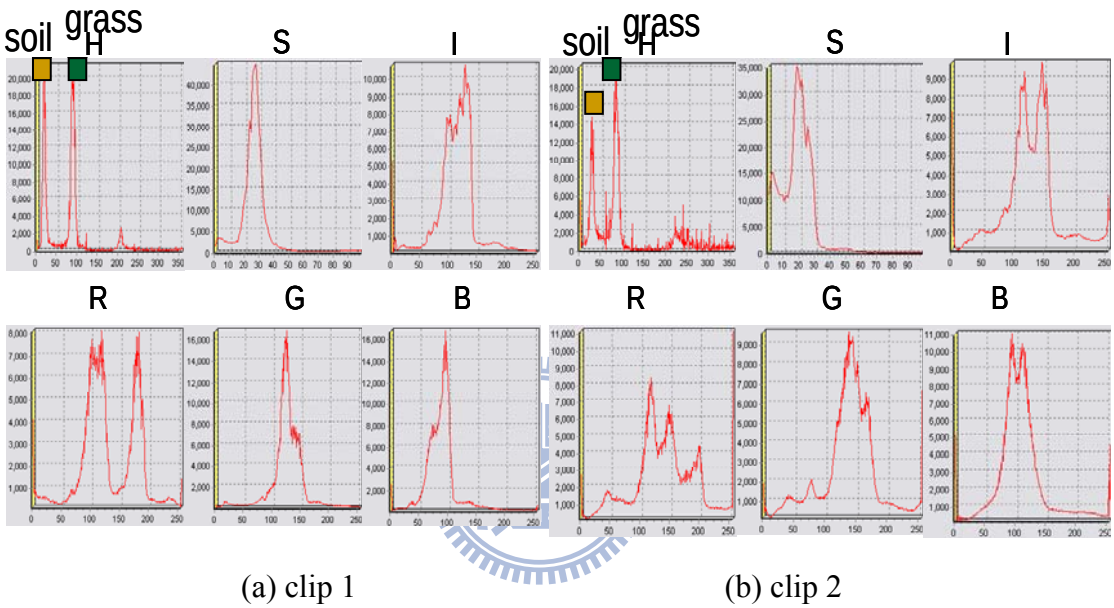
considerably many uninteresting segments in a baseball game video (e.g., commercials). Hence, some pre-processing schemes such as PC shot [1] and close-up shot detection are needed to trim out these segments.

As the section 2-1 described, a highlight is composed of some shots, each of which consists of several objects and features. To classify highlight, some tasks such as object and feature detection should be solved in the preliminary. In order to make the object detection easily, techniques of image processing are applied. An observation sequence is generated after feature extraction and object detection. Symbols mapped from feature vector by rule table are used as observations, and the number of states represented as shots are empirically determined. Then, we use Segmental K-means algorithm to create an initial HMM parameter λ and Baum-Welch algorithm to re-estimate highlight HMM parameter $\bar{\lambda}$. In classification step, an observations sequence is generated after feature extraction and object detection, too. Each event of baseball highlight HMM will output a probability and the best match will be chosen as the proposed answer. Extracted features and detected objects are used for frame classification by a rule table as listed in **Table 4-1** modified from [4] for the purpose of realizing the transition of shots. Details of the proposed approaches are described in the following sections. Section 4-2 introduces the color conversion. Section 4-3 describes object and feature detection. Section 4-4 describes frame classification. Section 4-5 and section 4-6 describe HMM learning and recognition of baseball event.

4.2 Color Conversion from RGB to HSI for Feature Extraction

In image processing or analysis, color is an important feature for our proposed object detection and feature (the percentage of grass and soil) extraction. However, the color of each baseball game in frames might vary because of the different angles of view and lighting conditions. To obtain the color distribution of grass and soil in

video frames, several baseball clips from different video source composed of grass and soil are input to produce the color histograms including RGB and HSI color space. **Fig. 4-2** takes two different baseball clips from different source as examples. Owing to the discrimination the Hue value in HSI color space is selected as the color feature, and the grass (green) and soil (brown) color range $[H_{a1},H_{b1}]$,$[H_{a2},H_{b2}]$ are set.



(a) clip 1 (b) clip 2

Fig. 4-2 The color space of RGB and HSI of two baseball clips.

After the grass and soil color range are set, the dominant color of green and brown is found. All colors except for green and brown are mapped to black as shown in **Fig. 4-3**.

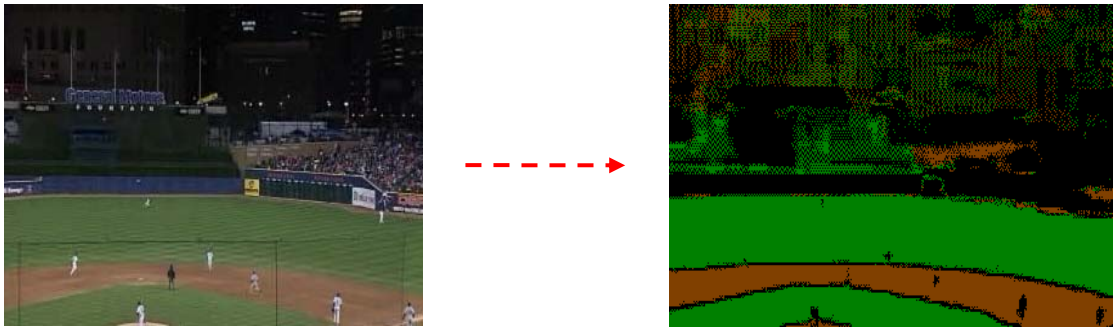
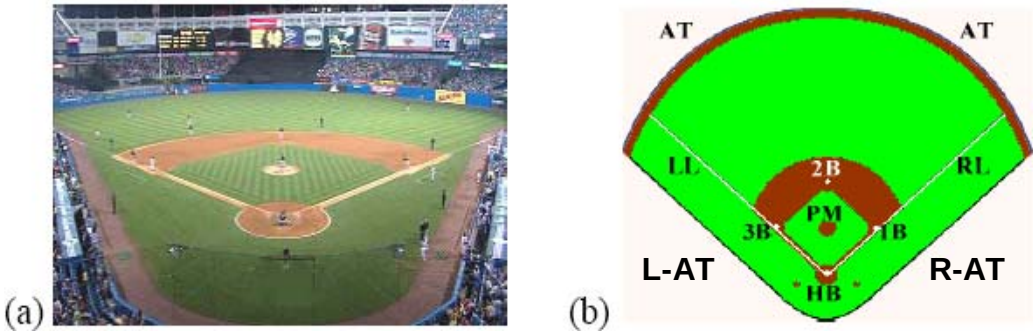


Fig. 4-3 The process of finding dominant colors.

4.3 Object (spatial) Detection

The baseball field is characterized by a well-defined layout of specific colors as described in Fig. 4-4. Furthermore, important lines and the bases are in white color, and auditorium (AT) is of high texture and no dominant color as shown in **Fig. 4-4(b)**.



(a) Full view of real baseball field (b) Illustration of baseball field

Figure 4-4 The field objects and features.

Each object will be elaborated as follows.

(1) Back auditorium (AT):

The top area which contains high texture and no dominant colors is considered as the auditorium, as the black area above the white horizontal line in Fig. 4-5(a).

(2) Left auditorium (L-AT) and right auditorium (R-AT):

The left area and right area which contains high texture and no dominant colors is considered as the left auditorium and right auditorium, as the left black area and the right black area marked with the white vertical line in **Fig. 4-5 (b)** and **Fig. 4-5 (c)**.

(a) (b) (c)

Fig. 4-5 Illustration of (a) back auditorium (b) left auditorium (c) right auditorium

(3) Left line (LL) and right line (RL) :

A Ransac algorithm, which finds the line parameter of line segments [12], is applied to the line pixels and then finds the left or right line. The line pixel is high intensity pixel greater than threshold σ_l excluding pixels in large white area and auditorium area. Either two pixels at a horizontal distance of $\pm\tau$ pixels or at a vertical distance of $\pm\tau$ pixels must be darker than σ_d, where $\sigma_d << \sigma_l$. **Fig. 4-6** shows the concept of excluding white area. The parameter τ should be set to approximately the double court line width. As illustrated in **Fig 4-6**, each square represents one pixel and the central one drawn in gray is a candidate pixel. Assuming that white lines are typically no wider than τ pixels (τ = 6 in our system), we check the four pixels, marked 'V' and 'H', at a distance of τ pixel away from the candidate pixel on the four directions. The central candidate pixel is identified as a white line pixel only if both pixels marked 'H' or both pixels marked 'V' are with lower brightness than the candidate pixel.

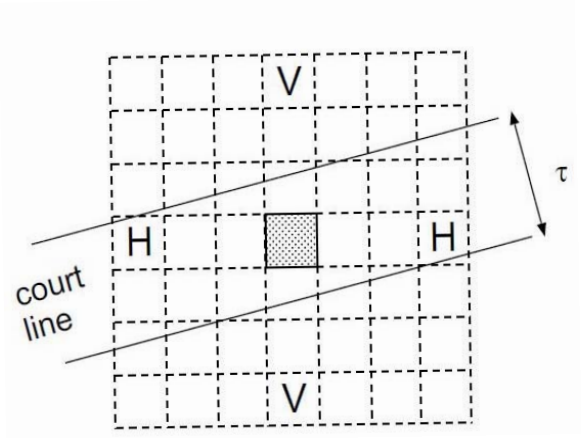
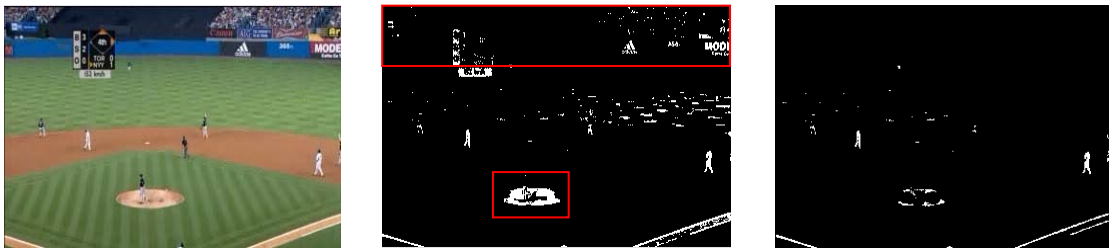


Fig. 4-6 Line pixel detection excluding large white area.

This process prevents that white pixels are extracted in large white areas including auditorium area or white uniforms. **Fig. 4-7** is an example of line pixel detection. **Fig. 4-7 (b)** shows that the intensity of pixels higher than a threshold of I component in HSI color space and **Fig. 4-7 (c)** shows that the remaining high intensity pixels after line pixel detection. In **Fig. 4-7 (c),** the high intensity pixels in pitch mound and

auditorium area is vanished.



(a) (b) (c)

Fig. 4-7 The result of retained high intensity pixel after line pixel detection algorithm: (a) original data (b) high intensity pixel before line pixel detection (c) high intensity data after line pixel detection

After line pixel detection, the Ransac algorithm is applied to find line parameter as shown in **Fig. 4-8**. In **Fig 4-8(a)**, the black point represents the high intensity pixel after the line pixel detection. In **Fig 4-8 (b)**, two points are randomly chosen and the red line is the line passing through the two points. In a specific area (the distance between the point and the line is less than 2, $\tau = 4$), the high intensity pixel will be accumulated denoted a score in this time. The action will repeat until the number of remaining high intensity pixel is less than N or the pre-defined iterative number is reached. Finally, the highest score and the line parameter will be stored as the proposed line in **Fig 4-8 (c)** and **Fig. 4-8 (d)**.

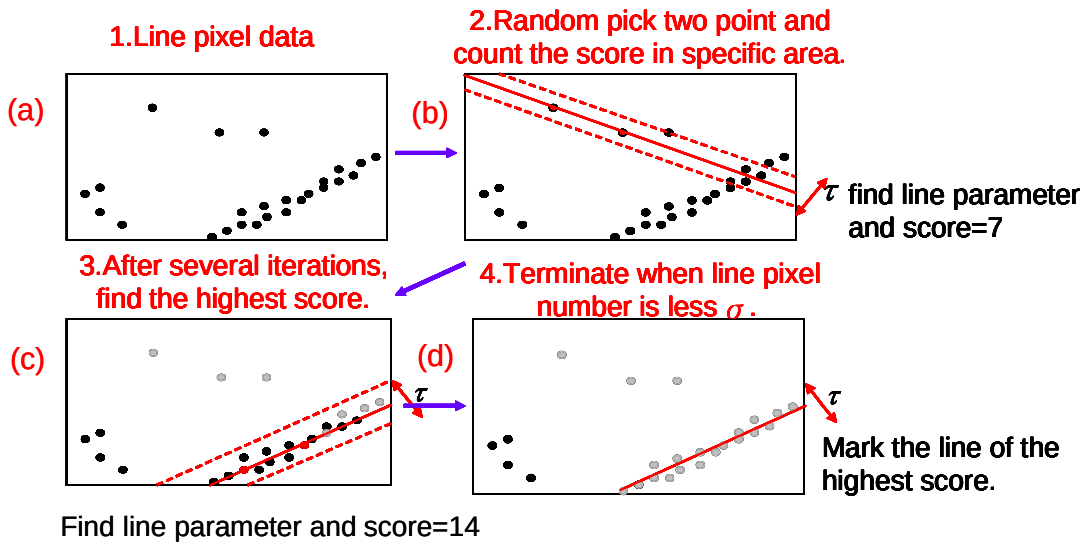


Fig. 4-8 Ransac algorithm for finding line parameter.

(4) Pitch mound (PM):

An ellipse soil region surrounded by a grass region would be recognized as pitcher's mound as shown in **Fig. 4-9**. Bounding box is applied to ellipse detection. The procedure is described as follows. (1) stop at a brown pixel, (2) find the upper, lower, left, and right bound stopped at the first green pixel in upper, lower, left, and right direction, (3) count the brown pixel percentage in the bounding box, and (4) the percentage in specific range will be considered a ellipse. Some illegal ellipse could fit the proportion of soil in bounding box. The illegal ellipse should be deleted to improve the frame classification and baseball event classification. A threshold of the brown pixel difference will be set in diagonal lines, upper and lower region, and right and left region as shown in **Fig. 4-10**. In **Fig. 4-10**, only **Fig. 4-10(c)** is a correct ellipse.

(5) First base (1B) and third base (3B):

The square region located on right line, if detected, in soil region would be identified as first base as shown in **Fig. 4-9**. Similarly, the square region located on left line, if detected, in soil region would be identified as third base.

(6) second base (2B):

In a soil region, a white square region on neither field line would be identified as second base as shown in **Fig. 4-9**.

(7) home base (HB):

Home base is located on the region of the intersection between left line and right line as shown in **Fig. 4-9**.

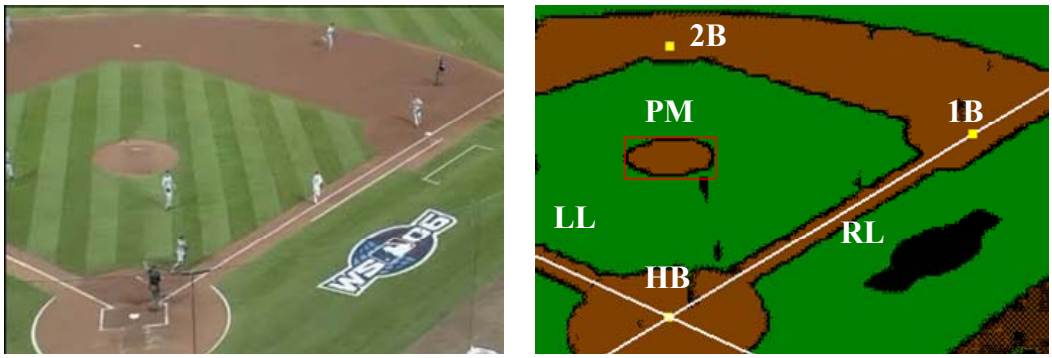


Fig. 4-9 Shows the objects of 1B, 2B, HB, LL, RL, and PM.

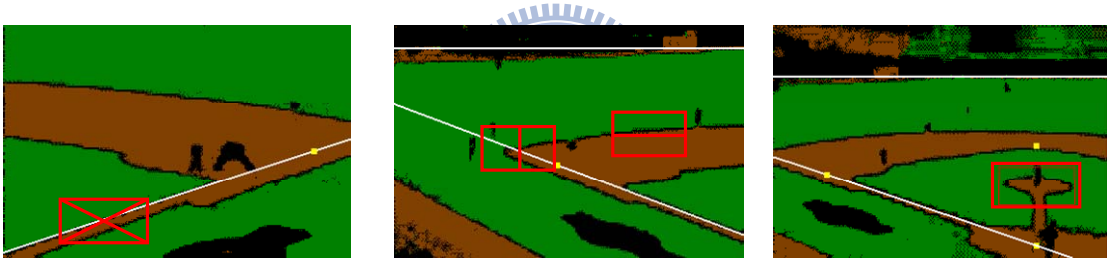


Fig. 4-10 Deletion of illegal ellipse

4.4 Frame Classification

Classification is divided into two orientations in this approach, one is frame type classification by using rule table as listed in **Table 4-1** modified from [4], and another is baseball event classification by using HMM. The classification of frame types can assist in realizing the shot transition or for other purposes. Sixteen frame types are defined and classified based on the position or percentage of some objects and features as described in section 4-3 in rule table. Sixteen typical region types is: IL (infield left), IC (infield center), IR (infield), B1 (first base), B2 (second base), B3 (third base), OL (outfield left), OC (outfield center), OR (outfield right), PS (play in

soil), PG (play in grass), AD (audience), RAD (right audience), LAD (left audience), CU (close-up), and TB (touch base), as shown in **Fig. 4-11**.

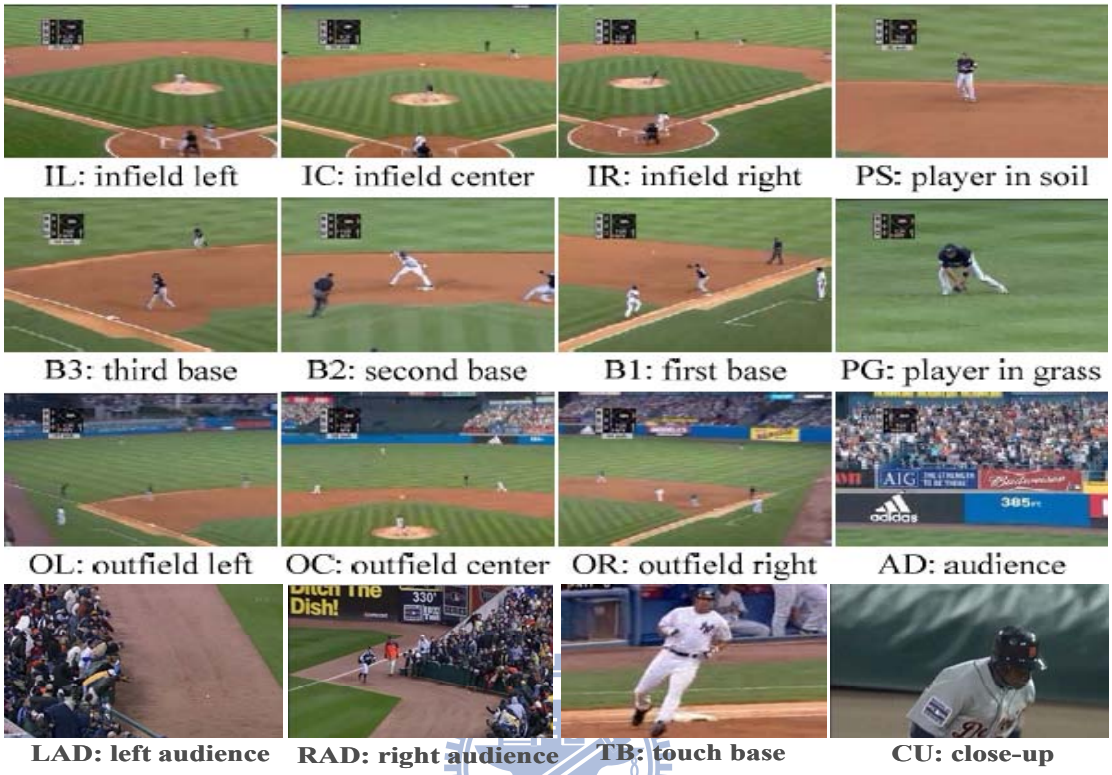


Fig. 4-11: Sixteen typical frame types

The rules of frame type classification are listed in **Table 4-1** modified from [4]**.** The symbols of first column are our sixteen defined frame type (**IR, IC**…) as shown in **Fig. 4-11**. W_f is the frame width, the function P(**Area**) return the percentage of the area **Area** in a frame, X(Obj) returns the x-coordinate of the center of the field object Obj, and W(Obj) returns true if the object Obj exists. Each field frame is classified into one of sixteen frame types using rule table (**Table 4-1** modified from [4]**)**. For example: a field frame would be classified as B1 frame type if the frame meets the following conditions: The percentage of AT is no more than 10%, the object of PM does not exist, the object of RL and 1B must exist, the percentage of soil is more than 30%.

IR:	$\{P(AT) \le 10\%,\ E(PM),\ X(PM) \le W_f / 3\} \|\|$ $\{P(AT) \le 10\%,\ \sim P(PM),\ E(RL),\ \sim E(1B)\} \|\|$ $\{P(AT) \le 10\%,\ \sim E(PM),\ E(RL),\ E(1B),\ P(soil) \le 30\%\}$
IC:	$\{P(AT) \le 10\%,\ E(PM),\ W_f < X(PM) \le W_f \times 2/3\} \|\|$ $\{P(AT) \le 10\%,\ \sim E(PM),\ \sim E(RL),\ \sim E(LL),\ E(2B),\ P(soil) \le 30\%\}$
IL:	$\{P(AT) \le 10\%,\ E(PM),\ X(PM) > W_f \times 2/3\} \|\|$ $\{P(AT) \le 10\%,\ \sim P(PM),\ E(LL),\ \sim E(3B)\} \|\|$ $\{P(AT) \le 10\%,\ \sim E(PM),\ E(LL),\ E(3B),\ P(soil) \le 30\%\}$
B1:	$\{P(AT) \le 10\%,\ \sim P(PM),\ E(RL),\ E(1B),\ P(soil) > 30\%\}$
B2:	$\{P(AT) \le 10\%,\ \sim P(PM),\ \sim E(LL),\ \sim E(RL),\ E(2B),\ P(soil) > 30\%\}$
B3:	$\{P(AT) \le 10\%,\ \sim P(PM),\ E(LL),\ E(3B),\ P(soil) > 30\%\}$
OR:	$\{10\% \le P(AT) < 75\%,\ E(PM),\ X(PM) \le W_f / 3\} \|\|$ $\{10\% \le P(AT) < 75\%,\ \sim P(PM),\ E(2B),\ X(2B) \le W_f / 3\} \|\|$ $\{10\% \le P(AT) < 75\%,\ \sim E(PM),\ \sim E(2B),\ E(RL),\ \sim E(LL)\}$
OC:	$\{10\% \le P(AT) < 75\%,\ E(PM),\ W_f < X(PM) \le W_f \times 2/3\} \|\|$ $\{10\% \le P(AT) < 75\%,\ \sim E(PM),\ E(2B),\ W_f < X(2B) \le W_f \times 2/3\}$
OL:	$\{10\% \le P(AT) < 75\%,\ E(PM),\ X(PM) > W_f \times 2/3\} \|\|$ $\{10\% \le P(AT) < 75\%,\ \sim P(PM),\ E(2B),\ X(2B) > W_f \times 2/3\} \|\|$ $\{10\% \le P(AT) < 75\%,\ \sim E(PM),\ E(2B),\ E(LL),\ \sim E(RL)\}$
PS:	$\{P(AT) \le 10\%,\ \sim E(PM),\ \sim E(2B),\ \sim E(RL),\ \sim E(LL),\ P(soil) > 30\%\}$
PG:	$\{10\% \le P(AT) < 75\%,\ \sim E(PM),\ \sim E(2B),\ \sim E(RL),\ \sim E(LL)\} \|\|$ $\{P(grass) > 60\%,\ \sim E(PM),\ \sim E(2B),\ \sim E(RL),\ \sim E(LL)\}$
AD:	$\{P(AT) > 75\%\}$
R-AD:	$\{10\% < P(R-AT) \le 50\%\}$
L-AD:	$\{10\% < P(L-AT) \le 50\%\}$
TB:	$\{10\% < P(AT) \le 50\%,\ E(RL)\ \&\&\ E(1B) \ \|\|\ E(LL)\ \&\&\ E(3B),\ P(soil) > 30\%\}$
CU	$\{P(AT) \le 10\%,\ \sim E(PM),\ \sim E(2B),\ \sim E(RL),\ \sim E(LL),\ P(black) \ge 50\%\}$
unknown	others

Table 4-1 Rules of frame type classification modified from [4].

Each frame is first recognized by the distribution of dominant color and white

pixels (intensity data). After object detection and looking up the rule table, we can know the detected objects and features in each frame and shot transition represented as an annotated string as shown in **Fig. 4-12**. The content of the sample field shot in **Fig.4-12** says that the ball is first batted into the left infield. Then, the shortstop picks up the ball and throws it to the first baseman. The batting process can be appropriately abstracted by the output string: **IL** (infield left) → **PS** (player in soil) → **B1** (first base). In order to filter out the misclassifications of frame types within a shot, some regular rules are applied. For example, the shot transition **IC → IR (or PS) → B1** appears frequently in the baseball event of ground out. When the current frame type is not IR or PS, but the previous frame type is IC and the next frame type is B1, the system will change the current type from incorrect type to IR. The regular rule is a set created previously from observation.

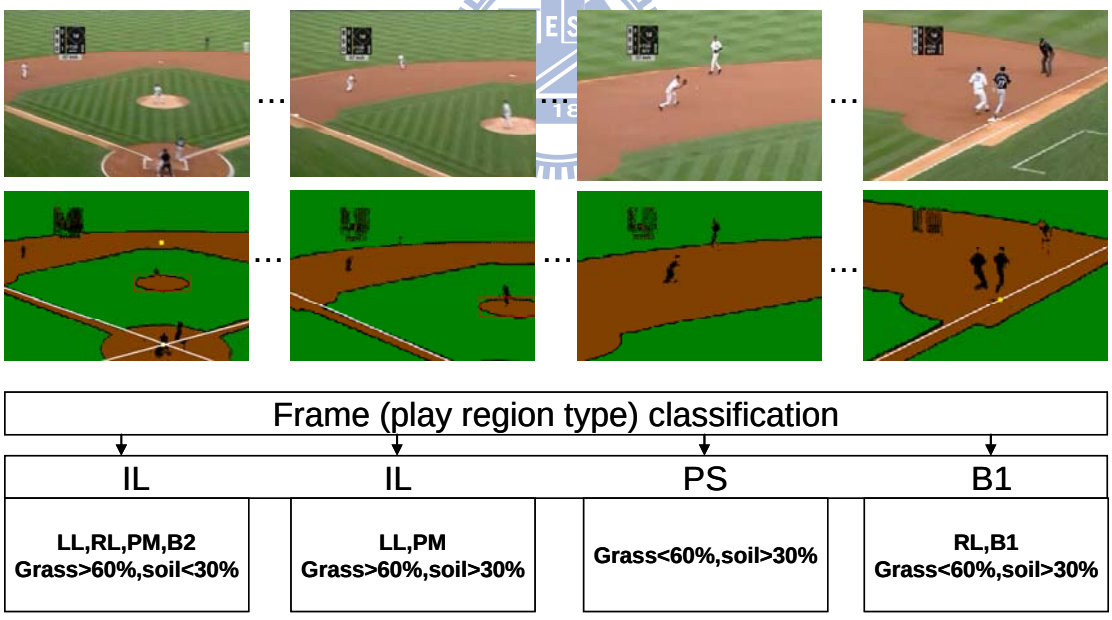


Fig. 4-12 Illustration of the annotated string of ground out example after frame classification

4.5 HMM Learning for Each Baseball Event

One HMM is created for each baseball event for recognizing time-sequential observed symbols. In our proposed method, twelve baseball events listed in **Table 4-2**

are defined so that there are twelve HMMs. Given a set of training data from each type of baseball event, we want to estimate the model parameters λ = (**A, B, π**) that best describe that each baseball event. First of all, Segmental K-means algorithm is used to create an initial HMM parameter λ and then Baum-Welch algorithm as described in section 3-3 is applied to re-estimate each HMM parameters $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ of baseball event.

Single	**Right foul ball**
Double	**Left foul ball**
Pop up	**Foul out**
Fly out	**Double play**
Ground out	**Home run**
Two-base out	**Home base out**

Table 4-2 List of twelve baseball events

In our proposed method, two features such as grass and soil, and ten objects as shown in **Fig. 4-4(b)** are used as observations represented as a 1×12 vector to record whether the object appears or not. To apply HMM to time-sequential video, the extracted features represented as a vector sequence must be transformed into a symbol sequence by rule table as listed in **Table 4-1** for later baseball event recognition. This is a well known technique, called vector quantization [17]. For vector quantization, codewords $g_j \in R^n$ represents an observation vector in the feature R^n space. Codeword g_j is assigned to symbol v_j. Consequently, the size of code book equals the number of HMM output symbols. Sixteen shots as shown in **Fig. 4-11** are viewed as hidden states.

Conventional implementation issues in HMM include (1) number of states, (2)

initialization, and (3) distribution of observation at each state. The first problem of determining the number of states is determined empirically and differs from each baseball event. The second problem can be approached by random initialization or using Segmental K-mean algorithm as described in section 3-3. Finally, the last problem can be solved by trying several models such as Gaussian model and choose the best one. In our approach, we choose Gaussian distribution. The following is the detailed description of each essential element.

State S: The number of states is selected empirically depending on different baseball event and each hidden state represents a shot type.

Observation O: the symbol mapped from rule table.

Observation distribution matrix B: use K-means algorithm and choose the Gaussian distribution at each state [15].

Transition probability matrix A: the state transition probability, which can be learned by Segmental K-means algorithm.

Initial state probability matrix π: the probability of occurrence of the first state, which is initialized by Segmental K-means algorithm after determining the number of states.

After determining the number of states and setting the initial tuple λ, to maximize the probability of the observation sequence given the model, we can use the Baum-Welch algorithm as described in section 3-3 to re-estimate the HMM parameter $\bar{\lambda}$. The initial probability, transition probability, output symbol distribution can be re-estimated by Eq. (30) (31) (32) and then replace initial tuple λ with $\bar{\lambda}$. A detail procedure is shown in **Table 4-3**:

Input: a set of observed symbol sequences (mapped by rule table) $O^1O^2...O^w$, and number of states are determined as input parameter.

Initialization: use Segmental K-means algorithm to compute initial λ and compute score: Score = $\sum_{1}^{w} P(O^w \mid \lambda)$

Repeat {

For each observed sequence O^w {

Using the given λ to calculate the following variable:

$\alpha_t(i)$ at each time t, state i using forward algorithm by Eq. (12) (13)

$\beta_t(i)$ at each time t, state i using backward algorithm by Eq. (16) (17)

$\gamma_t(i)$ at each time t, state i by Eq. (24)

$\varepsilon_t(i)$ at each time t, state transition from i to j by Eq. (26)

}

Calculate $\sum_{t=1}^{T}\gamma_t(i), \sum_{t=1}^{T-1}\gamma_t(i), \sum_{t=1}^{T-1}\varepsilon_t(i,j)$ by Eq. (27) (28) (29)

Re-estimate $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ by Eq. (30) (31) (32)

Score' = $\sum_{1}^{w} P(O^w \mid \bar{\lambda})$

If Score' < Score

Jump from Repeat loop

Else {

Score = Score'

$\lambda = \bar{\lambda}$

}

}

Table 4-3 HMM for baseball event learning

4.6 Baseball Event Recognition

The idea behind using the HMMs is to construct a model for each of the baseball event that we want to recognize. HMMs give a state based representation for each highlight. After training each baseball event model, we calculate the probability $P(\boldsymbol{O} \mid \lambda_i)$ of a given unknown symbol sequence $\boldsymbol{O}$ for each highlight model λ_i. We can then recognize the baseball event as being the one by the highest probable baseball event HMM.

Chapter 5
Experimental Result and Discussion

To test the performance of baseball event classification, we implement a system capable of recognizing twelve different types of baseball events. The test contains two parts: (1) frame type recognition (2) baseball event type recognition. The test data source can be divided into two groups, one is manual clips, and another is auto-segmented clips. In the first group, all clips are hand cut. In the second group, an ending point of clip is determined by detected close-up or specific shot. All video sources are Major League Baseball (MLB). 120 baseball clips from three different MLB video sources as training data and 122 baseball clips from two different MLB video sources as test data. The experimental result is shown in the following sections.

5.1 Frame Classification

The proposed frame classification system has been tested on Major League Baseball of broadcast baseball video. The categories to be recognized are sixteen frame types: 'infield left' (IL), 'infield center' (IC), 'infield right' (IR), 'first base' (B1), 'second base' (B2), 'third base' (B3), 'outfield left' (OL), 'outfield center' (OC), 'outfield right' (OR), 'audience' (AD), 'left audience' (L-AD), 'right audience' (R-AD), 'play in soil' (PS), 'play in grass' (PG), 'touch base' (TB). The video source was gotten from three different Major League Baseball games of broadcast video and digitized into 352×240 pixel resolution. The experimental result of frame type classification is in the **Table 5-1** and **Table 5-2**. **Table 5-1** shows the precision and recall of each frame type from clips manually clipped from broadcast video. **Table 5-2** shows the precision and recall of each frame type from clips automatically clipped from broadcast video.

Frame type	total	correct	false	Precision (%)	Recall (%)
IL	38	36	1	97.3	94.7
IR	91	90	3	96.8	98.9
IC	81	79	2	97.5	97.5
OL	51	47	2	95.6	92.2
OR	71	68	2	97.1	95.8
OC	67	61	5	92.4	91.0
B1	197	196	2	99.0	99.5
B2	48	39	5	88.6	81.3
B3	42	41	4	91.1	97.6
AD	72	72	1	98.6	100
TB	78	68	5	93.2	87.1
PG	377	366	21	94.5	97.1
PS	118	108	14	88.5	91.5
R-AD	69	69	0	100.0	100.0
L-AD	51	51	0	100.0	100.0
CU	95	89	6	93.7	93.7

Table 5-1 Recognition of frame types manually clipping

Frame type	total	correct	false	Precision (%)	Recall (%)
IL	61	55	2	96.5	90.2
IR	96	92	3	96.8	95.8
IC	81	79	3	96.3	97.5
OL	55	51	4	92.0	92.7
OR	76	73	2	97.3	96.1
OC	76	68	9	89.5	88.3
B1	318	301	6	98.0	94.7
B2	60	50	7	87.7	83.3
B3	42	41	4	91.1	97.6
AD	96	88	14	86.3	91.7
TB	83	68	5	93.2	81.9
PG	509	485	44	91.7	95.3
PS	168	155	29	84.2	92.3
R-AD	81	81	4	95.3	100
L-AD	73	73	5	93.6	100
CU	117	109	11	90.8	93.2

Table 5-2 Recognition of frame types automatically clipping

The second column “total” represents the total number of field shots containing the frame type designated in the first column. Note that a field shot might comprise more than one frame type. The “correct” and “false” represent the number of correct detections and false alarms. Both the precision and recall are about 90% except for the precision of B2 (second base), AD (audience), PS (player in soil) and the recall of OC (outfield center), B2 (second base), TB (touch base).

The low recall rate of frame type B2 and OC might result from the missed detection of field object *2B*. The low recall rate of frame type TB (touch base) might be that TB and AD are similar because both TB and AD have large area of auditorium and we don’t detect the object of player in this proposed method. The low precision rate of frame type B2 might result from the incorrect detection of field object *2B*. Sometimes the ball on soil is detected as a 2B (second base). The low precision rate of frame type AD (audience) might result from the incorrect classification between AD and TB. The low precision rate of frame type PS (play in soil) might result from the incorrect classification between PS and B2 (sometimes a frame of miss detection of object 2B will be classified into frame type of PS). These could be improved by enhancing field object detection and refining the rules of frame type classification. Overall, we achieve good performance about 90%.

5.2 Baseball Event Classification

The proposed baseball event classification system has been tested on Major League Baseball of broadcast baseball video. The categories to be recognized are twelve types of baseball events: *single, double, pop up, fly out, foul out, ground out, two-base out, right foul ball, left foul ball, double play, home run, and home base out* as described in **Table 4-2**. The video source was gotten from three different Major League Baseball games of broadcast video and digitized into 352×240 pixel resolution. The experimental result of baseball event type classification is in the

Table 5-3 and **Table 5-4**. **Table 5-3** shows the precision and recall of clips manually clipped from broadcast videos. **Table 5-4** shows the precision and recall of clips automatically clipped from broadcast video.

Event type	Total	Correct	false	Precision (%)	Recall (%)
Single	25	20	1	95.2	80.0
double	8	2	1	66.7	25.0
Pop up	7	7	2	77.8	100.0
Fly out	22	21	4	84.0	95.5
Foul out	1	1	0	100.0	100.0
Ground out	29	27	3	90.0	93.1
Two-base out	4	4	0	100.0	100.0
Right foul ball	12	12	0	100.0	100.0
Left foul ball	6	6	2	75.0	100.0
Double play	1	1	2	33.3	100
Home run	6	5	1	85.7	83.3
Home base out	1	1	0	100.0	100.0
Total	122	106			87.3

Table 5-3 Recognition of baseball events manually clipping

Event type	Total	Correct	false	Precision (%)	Recall (%)
Single	25	20	3	87.0	80.0
double	8	2	1	66.7	25.0
Pop up	7	7	2	77.8	100.0
Fly out	22	19	5	86.4	79.2
Foul out	1	1	0	100.0	100.0
Ground out	29	27	4	87.1	93.1
Two-base out	4	2	0	100.0	50.0
Right foul ball	12	12	0	100.0	100.0
Left foul ball	6	6	2	75.0	100.0
Double play	1	1	2	33.3	100.0
Home run	6	5	1	85.7	83.3
Home base out	1	1	0	100.0	100.0
Total	122	102			84.4

Table 5-4 Recognition of baseball events automatically clipping

Both the precision and recall are about 80% except for the precision of *double*, *double play* and the recall of *double*, *two-base out*. The low recall rate of baseball event *double* and *two-base out* might result from the missed detection of field object *2B*. The low precision rate of baseball event *double* might be that the transitions of *double* and *Home run* are similar if the batter hits the ball to the audience wall.

The low precision rate of baseball event *double play* might be that the transitions of *double play* and *ground out* are similar if the batter hits the ball around the second base as shown in **Fig. 5-1**. **Fig. 5-2** shows the miss detection of *right foul ball* and *home run* due to the similar shot transition. **Fig. 5-3** shows some ambiguities in nature of baseball events such as *ground out* and *left foul ball* even if those baseball events are judged by people. **Fig. 5-4** shows miss detection between *single* and *ground out* because the player in first base does not catch the ball and we do not detect the ball object.

The miss detection of highlights can be classified into four reasons: (1) similar shot transition, (2) miss object detection, (3) detected objects are not enough, and (4) ambiguity in nature. These could be improved by detecting the object of ball and players, or add additional information such as scoreboard information. Overall, we still achieve good performance.

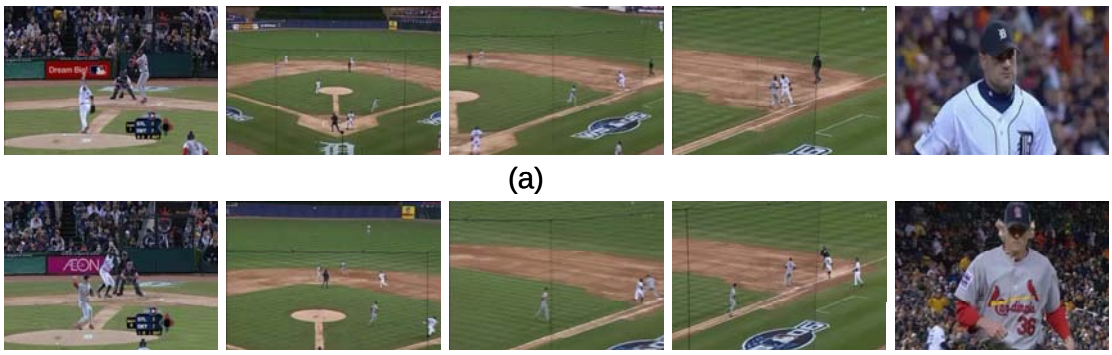


Fig. 5-1 Comparison between (a) ground out and (b) double play

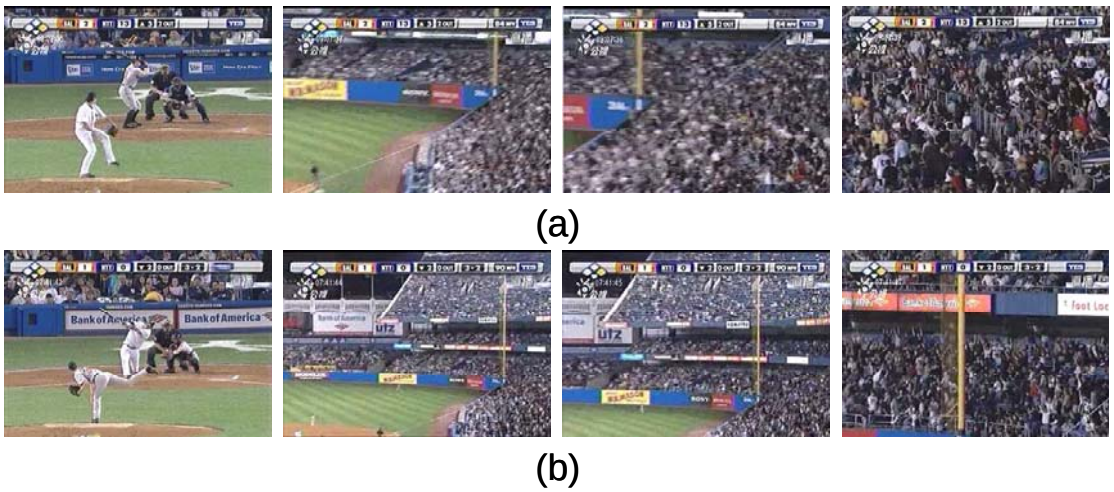


Fig. 5-2 Comparison between (a) right foul ball and (b) home run

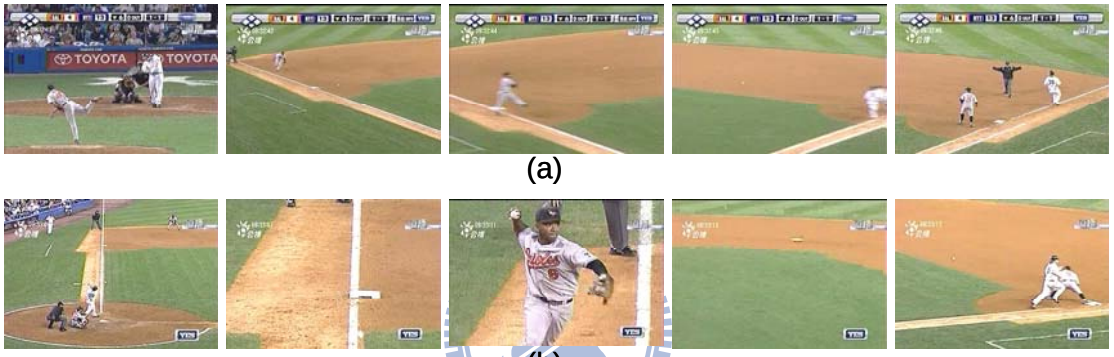


Fig. 5-3 Ambiguity of (a) left foul ball (b) replay of left foul ball

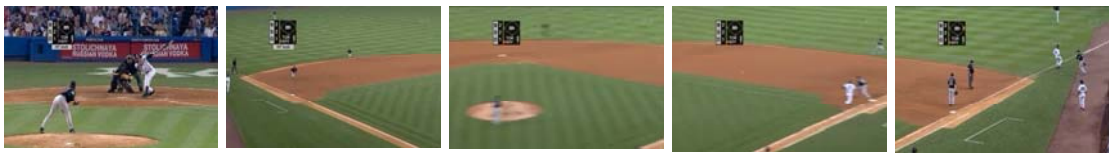


Fig. 5-4 Ambiguity of ground out and single

5.3 Other Discussions

In the line detection in section 4.3, lines are detected by Ransac algorithm rather than Hough transform. Hough transform can detect most of lines in a frame, but the time complexity is too complicated. Ransac algorithm can find “apparent” lines. The time complexity depends on the number of line pixels. In baseball game, the most important lines are left line and right line. The left line or right line in each frame of baseball game is marked with white color. The number of lines is not too many and

the lines are apparent in baseball field. According to the observation, we adopt Ransac algorithm to do line detection. The algorithm is applied in other sports such as tennis [12].

Second base detection in section 4.3, some conditions will be viewed as second base as shown in **Fig. 5-5**.

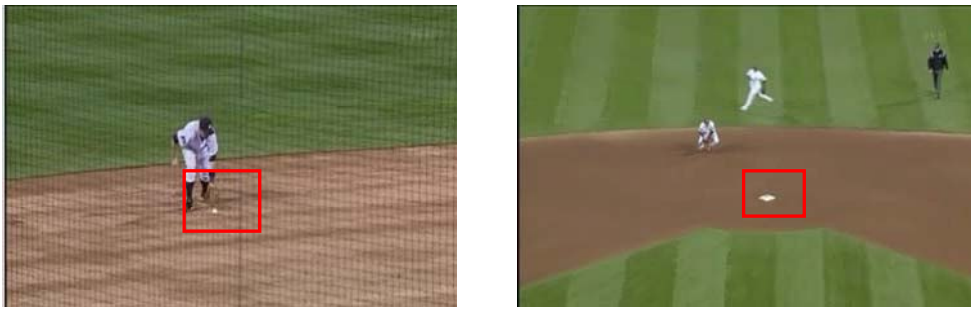


Fig. 5-5 Ambiguity in second base

We should set a threshold of region size of second base, but it is very difficult to set the boundary because the size of second base varies in each frame due to the zooming of the camera. It is also the main reason of poor detection in B2 frame type classification.

Determining the type of Markov model is also an important issue in our system. The original Hidden Markov model is shown in **Fig. 5-6**. At the same time, the complicated model can be simplified to left-to-right Hidden Markov model as shown in **Fig. 5-7** and the model is applied extensively in speech recognition. In baseball game, baseball events are usually composed of several shots and each baseball event has specific shot transition. Take four baseball events in **Fig. 2-4** [5] as an example. Some phenomena can be exploited such as the baseball event of Home run: The state of Audience View and Running Closeup are bi-directed. So, original Hidden Markov Model is adopted in our system for baseball event classification.

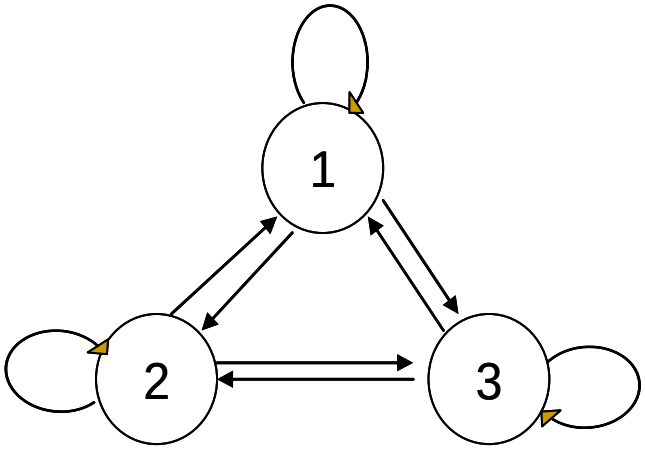


Fig. 5-6 3-state original Hidden Markov Model

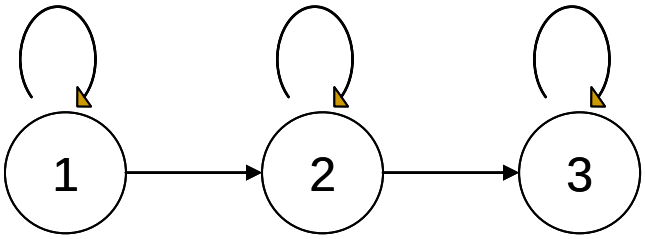
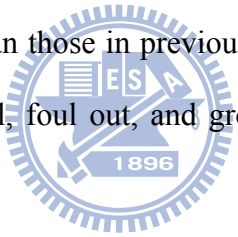


Fig. 5-7 3-state left-to-right Hidden Markov Model

All discussions above, we know the current problems in the proposed system. The detail of how to improve the problems will be described in chapter 6.

Chapter 6
Conclusion and Future Work

We can achieve good performance of baseball event classification due to the high precision in object detection and varieties of frame types. High precision in object detection makes the precision of frame type classification higher and increases varieties of frame types and baseball event types. High precision in frame type classification of R-AT frame type and L-AT frame type makes the precisions of *foul ball* and *foul out* baseball event types higher. High precision in object detection of bases and lines, frame type classification of infield types and outfield types makes the precisions of *ground out* and *fly out* baseball event types higher than those in previous work. All points above, some viewpoints are concluded in our system: (1) the frame classification achieves a high precision which is about 90%, (2) the hitting baseball event types are detected more than those in previous works, and (3) the precisions in classification of fly out, foul ball, foul out, and ground out are better than those in previous works.



However, some problems as shown in **Fig. 5-1**, **Fig. 5-2**, **Fig. 5-3**, and **Fig. 5-4** in section 5-2 can be improved by adding some intelligent information such as scoreboard information. Take *double play* and *ground out* as an example. We can recognize the two different baseball events from the #out information in scoreboard even if both shot transitions are similar. Furthermore, we hope that the baseball event type classification in proposed system is robust, but the current system just adapts to MLB video sources. All problems above, our future works are as follows. (1) Add scoreboard information to solve some ambiguities of baseball events, (2) Add player or ball tracking to raise the correctness rate of baseball event classification, and (3) Increase varieties of baseball video sources such as Nippon Professional Baseball (NPB) as training data to make the proposed system more robust.

Bibliography

[1] M. Kumano, Y. Ariki, K. Tsukada, S. Hamaguchi, and H. Kiyose, "Automatic Extraction of PC Scenes Based on Feature Mining for a Real Time Delivery System of Baseball Highlight Scenes," *IEEE international Conference on Multimedia and Expo, vol. 1, pp. 277-280, 2004.*

[2] H. S. Chen, H. T. Chen, W. J. Tsai, S. Y. Lee, and J. Y. Yu, "Pitch-by-Pitch Extraction from Single View Baseball Video Sequences," *IEEE International Conference on Multimedia and Expo, pp. 1423-1426, 2007.*

[3] W. Hua, M. Han, and Y. Gong, "Baseball scene classification using multimedia features," *IEEE International Conference on Multimedia and Expo, vol. 1, pp. 821-824, 2002.*

[4] H. T. Chen, M. H. Hsiao, H. S. Chen, W. J. Tsai, and S. Y. Lee, "A baseball exploration system using spatial pattern recognition," *Proc. IEEE International Symposium on Circuits and Systems, pp.3522-3525, May 2008.*

[5] P. C. , M. H., and Y. Gong, "Extract Highlight From Baseball Game Video With Hidden Markov Models," *International Conference on Image Processing, vol. 1, pp.609-612, 2002.*

[6] T. Mochizuki, M. Tadenuma and N. Yagi, "Baseball Video Indexing Using Patternization Of scenes and Hidden Markov Model," *IEEE International Conference on Image Processing, vol. 3, pp. III -1212-15, 2005.*

[7] N. H. Bach, K. Shinoda and S. Furui, "Robust Highlight Extraction Using Multi-stream Hidden Markov Model For Baseball Video," *IEEE International Conference on Image Processing, vol. 3, pp. III- 173-6, 2005.*

[8] M. Fleischman, B. Roy and D. Roy, "Temporal Feature Induction for Baseball Highlight Classification," *Proc. ACM Multimedia Conference, Augsburg, Germany, pp. 333-336, 2007.*

[9] M. H. Hung, C. H. Hsieh and C. M. Kuo, "Rule-based Event Detection of Broadcast Baseball Videos Using Mid-level Cues," *in Proceedings of IEEE International Conference on Innovative Computing Information and Control, pp. 240-244, 2007.*

[10] C. H. Liang, W. T. Chu, J. H. Kuo, J. L. Wu, and W. H. Cheng, "Baseball event detection using game-specific feature sets and rules," *in Proceedings of IEEE*

International Symposium on Circuits and Systems, vol. 4, pp.3829-3832, 2005.

[11] M. H. Hung and C. H. Hsieh, "Event Detection of Broadcast Baseball Videos," *IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, no.12, pp.1713-1726, 2008.*

[12] D. Farin, J. Han and H. N. Peter, "Fast Camera Calibration for the Analysis of Sport Sequences," *IEEE International Conference on Multimedia & Expo, pp. 482–485, 2005.*

[13] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE Signal Processing Magazine, vol. 3, no. 1, pp4-16, 1986.*

[14] G.D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE, vol. 61, no. 3, pp. 263-278, 1973.*

[15] B. H. Juang and L. R. Rabiner, "The segmental K-means algorithm for estimating the parameters of hidden Markov models," *IEEE Transaction of Acoustics Speech and Signal Processing, vol. 38, no. 9, pp. 1639-1641, Sept. 1990.*

[16] B. H. Juang and L. R. Rabiner, "A probabilistic distance measure between HMMs," *AT&T Technical Journal, vol. 64, no. 2, pp. 391-408, 1985.*

[17] X. D. Huang, Y. Ariki, and M. A. Jack, "Hidden Markov Models for Speech Recognition," *Edingurgh Univ. Press, 1990.*

[18] L. R. Rabiner and B. H. Juang, "A Tutorial on Hidden Markov Models and selected Applications in Speech Recognition," *Proceedings of the IEEE, pp. 257-286, 1989.*

[19] H. T. Chen, M. C. Tien, Y. W. Chen, W. J. Tsai and S. Y. Lee, "Physics-based ball tracking and 3D trajectory reconstruction with applications to shooting location estimation in basketball video," *Journal of Visual Communication and Image Representation, vol. 20, no. 3, pp. 204-216, 2009.*

[20] Hua-Tsung Chen, Hsuan-Sheng Chen, Ming-Ho Hsiao, Wen-Jiin Tsai and Suh-Yin Lee, "A Trajectory-Based Ball Tracking Framework with Visual Enrichment for Broadcast Baseball Videos", *Journal of Information Science and Engineering (JISE), vol. 24, no. 1, pp.143-157, Jan 2008.*