A study of automatic page turning of musical scores by detecting player's nods

Haruka JIBIKI[*1], Toshikazu SHIMIZU[*1], Yasuyuki SAITO[*1],
Eita NAKAMURA[*2], Shigeki SAGAYAMA[*2]

*1 Dept. of Information Engineering, National Institute of Technology, Kisarazu College

*2 School of Interdisciplinary Mathematical Sciences, Dept. of Frontier Media Science, Meiji University

In this study, we construct a page turning system for musical scores in PDF files on a computer. Piano players often ask a page turner to turn over pages of a musical score, especially in case of that, the music piece is complex or its designated tempo is fast. However, it is necessary to turn pages by oneself during practice. Turning pages by oneself is problematic with incomplete turnings and misses, because it should be done very hurriedly. Piano players use both hands and feet. Thus, the only body part which can be used is head, and in fact they usually indicate a turning to the page turner by nods. We pay attention to this action and develop software to detect it by using USB cameras. Moreover, we solve the face detection problem by using a hidden Markov model.

1. Introduction

In general, an instrument player turns over musical scores by oneself. In contrast, a piano player asks a page turner to turn over musical scores. However, it is necessary to turn the pages by oneself during practice, even if the player plays long and complex music with both hands at the same time without a break.

A German company Forma Bene developed an automatic page turning device[1)] called “Volta Bene” (Fig.1). The device has an arm with a magnet and a foot switch and works on battery. The user puts iron chips on the musical score in advance, and stamps the foot switch to turn over a page. The arm swings like a car wiper, and the magnet attracts the iron chip. Then the musical score is turned over. Although it has an advantage that the user can use real musical scores during the practice, the device has many problems: (1) it often fails to turn over, (2) external power supply is necessary for stable performance, (3) the player can not stamp the foot switch when he performs the piano using pedal with both feet at the same time, and so on.

Actually, piano players sign a human page turner by nods. Therefore, we focus on the nods to construct an automatic page turning system for the musical score.

Fig.1 Mechanical automatic turning over system for musical score “Volta Bene”

2. A method of turning over musical scores

2.1 Face tracking

There are some devices to detect player’s nods; for example, USB camera, RGB-D camera, acceleration sensor. Lately, it has been possible to produce RGB-D camera and acceleration sensor at relatively low cost, and they are commercially available at affordable prices. However, in general, they have not been yet integrated in laptops and mobile devices such as tablet terminal, smart-phone, and so on. Moreover, in the case of using the acceleration sensor, the user must put it on the head, which is not natural. On the other hand, USB camera is common now and equipped with standard laptops and mobile devices. Therefore, we use it in this study.

We have designed a face tracking system consisting of two stages. In the first stage, the system detects face region by using Haar-like characteristics in the whole image obtained by the USB camera.

Next, the system extracts the central region of the input image as a nose region. In comparison, the nose region does not deform significantly. Moreover, we assume that the piano player is in front of the piano, and the distance between the player and the piano is limited within a certain range. Thus the size of the apparent nose does not change significantly. After this, to realize the high speed nose detection, the system focuses on a periphery of the nose region instead of the whole image.

*1 2-11-1 Kiyomidai-higashi, Kisarazu, Chiba, Japan
e-mail: saito@j.kisarazu.ac.jp

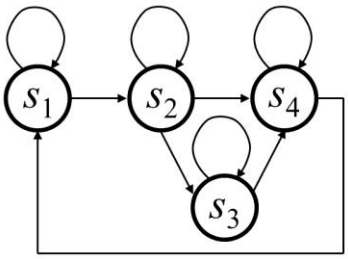

Fig.2 Designed hidden Markov model.

2.2 Player's nod estimation

To avoid effects of noises and player's head motions which are not subject to detect, we introduce a hidden Markov model (HMM) as follows.

We divide a player's nod into following sections: (1) stationary, (2) moving down, (3) stationary, (4) moving up, (5) stationary. We regard each section as a "state" and consider that the time series of the nod actions are output (see below) while transiting between the states. In this case, although only the outputs can be observed, it is difficult to know directly whether they are output from which state. Thus, the state transition is hidden. Therefore, we introduce HMM which is known that it is suitable to model such time series. Suppose the first and last "stationary" states are same, we design an HMM as shown in Fig.2. Each state s_j corresponds to each section of nod action: "stationary" (s_1), "moving down" (s_2), "stationary" (s_3; may be skipped), and "moving up" (s_4). Each state transits to itself or next state stochastically.

Consider that each state outputs a distance d_p which indicates a difference of between current and previous nose position of vertical direction as a nod action. Moreover, we also consider distinguishing a player's nod from just looking down, for example, the player looks at the key of the piano. This problem is solved by counting up the time length of the state transition which stays on s_3. However, when a predetermined time elapses while the nose stays almost the same position (namely, staying s_1 or s_3), all the values of d_p in the buffer become close to zero because the buffer which stores the nose positions consists of a finite length. Thus, it is difficult to distinguish whether the transition state is s_1 or s_3 only using the distance d_p. Therefore, we introduce one other output d_a from the states which indicates the distance of between the current and the average nose position of vertical direction. If the nose stays at lower position, d_a becomes a large value compared to staying at the initial position. And then, the values used for the update calculation of the average are vertical coordinates of the nose positions which are obtained only in time in which the estimated transition state is s_1.

Table 1 Symbols using Viterbi algorithm

Symbol	Description
c	Number of conditions
n	Number of time series
$s_j^{(t)}$	State at certain time t (j=1,2,…, c)
$o^{(t)}$	Output
r_i	Initial selected probability of state i (i = 1, 2, …,c)
a_{ij}	Transition probability from state i to j
$b(s_j^{(t)},o^{(t)})$	Probability that output $o^{(t)}$ is observed from the state $s_j^{(t)}$.
$p_j^{(t)}$	Maximum probability on state j at t
$q_j^{(t)}$	Storage for representation of optimum state series
$z^{(t)}$	State number of optimum series

Although we are not able to know the past transited state clearly, Viterbi algorithm is well known as an efficient solution of state transition estimation of the HMM. The detail of Viterbi algorithm is shown in section 3. At first, we determine the HMM paths of player's nods. If a player performs the nod, the order of the estimated state transition should be $s_1\ s_2\ s_4\ s_1$. Or, in the case of the order $s_1\ s_2\ s_3\ s_4\ s_1$, the staying time of s_3 is short. In fact, because the HMM path includes self transitions, the staying time length at each state is various. Hence the path matching is done after the time compression as reducing some unchanged states, for example, the time series $s_1\ s_2\ s_2\ s_2\ s_3\ s_3\ s_4\ s_4\ s_4\ s_1$ is compressed into $s_1\ s_2\ s_3\ s_4\ s_1$. When time series of both d_p and d_a from analysis of the USB camera image are given and the estimation results of Viterbi algorithm agree with above paths, the page turning sign is issued.

3. Viterbi algorithm

The Viterbi algorithm can estimate the optimum path of state transition of time series[2]. It is based on dynamic programming. Mathematical symbols which are used in the Viterbi algorithm are shown in Table 1.

The Viterbi algorithm consists of four steps as follows:

Step 1 Initialization

$$p_i^{(1)} = r_i\, b(s_i^{(1)}, o^{(1)}) \quad (1)$$

$$q_i^{(1)} = 0 \quad (2)$$

$(i = 1, 2, \ldots, c)$

Step 2 Recursive calculation

$$p_j^{(t)} = \max_i \{ p_j^{(t\text{-}1)}\, a_{ij} \}\, b(s_j^{(t)}, o^{(t)}) \quad (3)$$

$$q_j^{(t)} = \mathrm{argmax}_i \{ p_i^{(t\text{-}1)}\, a_{ij} \} \quad (4)$$

$(t = 2, 3, \ldots, n) \quad (i, j = 1, 2, \ldots, c)$

Step 3 Termination

$$z^{(n)} = \mathrm{argmax}_i \{ p_i^{(n)} \} \qquad (i = 1, 2, \ldots, c) \quad (5)$$

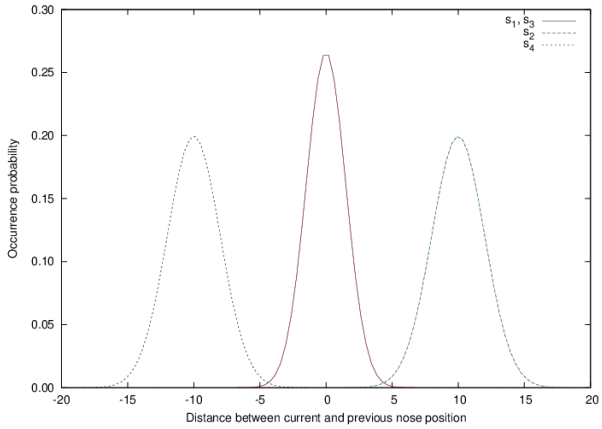

Fig.4 Output occurrence probability $b(s_{t,i}, d_p)$

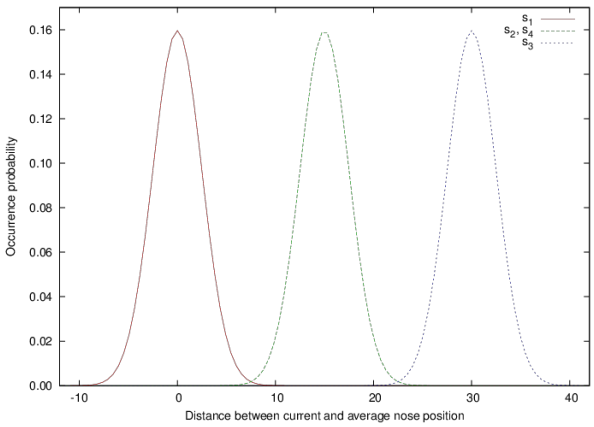

Fig.5 Output occurrence probability $b(s_{t,i}, d_a)$

Step 4 Representation of state series

$$z^{(t)} = q_{z(t+1)}^{(t+1)} \quad (t = n\text{-}1, \ldots, 2, 1) \tag{6}$$

Note that the order of the estimating is reverse for t.

In case that all r_i are set to non-zero probabilities, all states are possible to become initial state. On the other hand, if some of r_i are given zero arbitrarily, the specified states are only able to be initial states. Thus, it is possible to control the path of the state transition to some extent.

4. Implementation

We implement the system by using Java[3]. Java is designed based on the concept of object-oriented programming, and to reduce the complexity of software development and maintenance, to increase the development efficiency and maintainability. Java is excellent in versatility, and it is possible to perform the development and deployment of application software platform-independent. In this study, although we develop the system on a laptop, we will apply it on mobile devices.

We use OpenCV[4] for Java which is developed and released by Intel. It is a computer vision library of open source. Currently, the development of the OpenCV has been taken over by Willow Garage. The OpenCV is very useful, and can be easily used. It has many methods, and they work very quickly.

We introduce PDFRenderer[5] which is a presentation library for PDF file. A method `RePaintDisplay()` expresses a page of a PDF file. The argument of the method is a page number. Suppose a class variable `nowPageNum`s indicate page number, we can use the following method to display the next page:

```
RePaintDisplay(++nowPageNum);
```

Thus, when the nod estimation system detects player's nods, above method with the argument is called.

5. Experimental results

We used a laptop whose CPU is Intel Core i5 1.6 GHz and memory is 8 GB, and PDF musical scores which are not processed such as embedding special symbols.

First, we defined initial probabilities r_i, state transition probabilities a_{ij} and output probabilities $b(s_j^{(t)},o^{(t)})$ as follows:

$$r_i = \begin{pmatrix} 0.5 & 0.0 & 0.5 & 0.0 \end{pmatrix}, \tag{7}$$

$$a_{ij} = \begin{pmatrix} 0.8 & 0.2 & 0.0 & 0.0 \\ 0.0 & 0.6 & 0.2 & 0.2 \\ 0.0 & 0.0 & 0.8 & 0.2 \\ 0.2 & 0.0 & 0.0 & 0.8 \end{pmatrix}, \tag{8}$$

$$b(s_j, o^{(t)}) = b(s_j, d_p)\, b(s_j, d_a), \tag{9}$$

$$b(s_j, d_p) = \frac{1}{\sqrt{2\pi\sigma_{p,j}^2}} \exp\left(-\frac{(d_p-\mu_{p,j})^2}{2\sigma_{p,j}^2}\right), \tag{10}$$

$$b(s_j, d_a) = \frac{1}{\sqrt{2\pi\sigma_{a,j}^2}} \exp\left(-\frac{(d_a-\mu_{a,j})^2}{2\sigma_{a,j}^2}\right), \tag{11}$$

where d_p is the distance between current and previous nose position, d_a is the distance between current and average nose position. And we assumed parameters μ and σ as follows:

$$\mu_p = \{0.0, 10.0, 0.0, -10.0\}, \tag{12}$$

$$\sigma_{p,j}^2 = \{1.5, 2.0, 1.5, 2.0\}, \tag{13}$$

$$\mu_a = \{0.0, 15.0, 30.0, 15.0\}, \tag{14}$$

$$\sigma_{a,j}^2 = \{1.5, 2.0, 1.5, 2.0\}, \tag{15}$$

thus, $b(s_j^{(t)}, d_p)$ and $b(s_j^{(t)}, d_a)$ are shown in Fig.4 and Fig.5.

As the result of experiments for a professional piano player, the system turned over all the pages accurately. Fig.6 shows a result of the player's nose position. Note that the axes values are set according to image coordinates. From this figure, it seems that when a line of sight moves from top to bottom of musical score, the nose turns downward a little bit slowly. If the player would like to turn over the page, the face turns down quickly and greatly. On the other hand, in the case of looking down simply, for example, the player looks at the keyboard, the nose position stays in the bottom.

Fig.7 shows the other result which includes just looking down actions. In this figure, the time sections between 13 and 29, 41 and 64, 89 and 107 indicate their

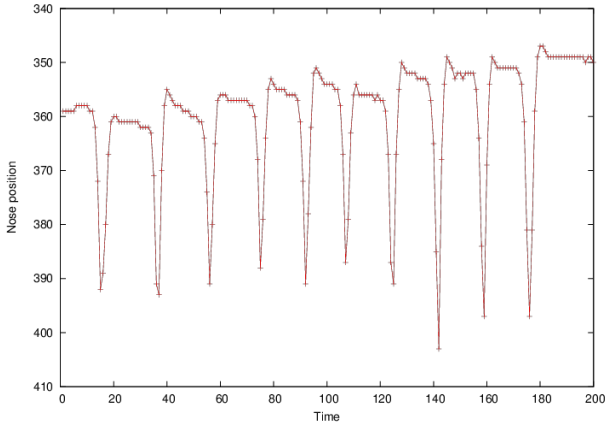

Fig.6 Transtion of player's nose position

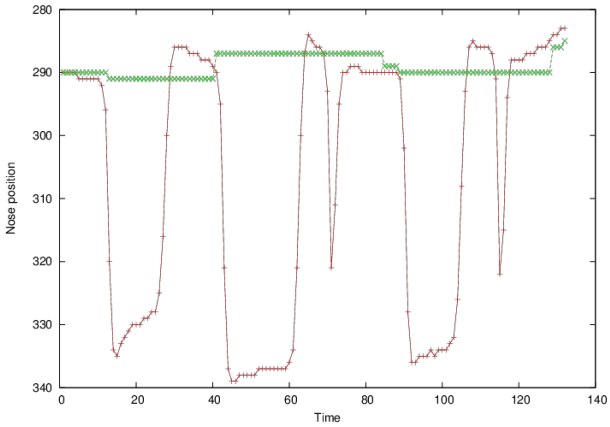

Fig.7 Transition of player's nods which includes just looking down actions. Red line indicates nose position, green line demonstrates average nose position

actions. We confirmed that the page turning did not occur during these time sections. Its process is valid. In addition, the average nose position does not change in these time sections. On the other hand, the time sections between 64 and 74, 113 and 118 demonstrate player's nods. In these cases, the page turning occurred properly.

6. Discussion

6.1 Aggressive use of horizontal head motions

In this study, horizontal head motions (clockwise and counterclockwise rotations with respect to the vertical axis of the head) are ignored because they do not contribute to nods. If it is allowed to use some special head motions, some functions are able to be realized. For example, we can relate horizontal head motions to turning to the previous pages, and to jumping to the coda page. Although this method is useful, such head motions are not usually used for real piano performance. Therefore, it is necessary to verify the effectiveness by using this method in practice.

6.2 Combination of our method and fully automatic page turning system

Our system can automatically turn over the pages. However, it can be also said to be a kind of manual operation software because the user instructs it by nods. On the other hand, theoretically, fully automatic page turning system will be realized if player's performance position in the music is recognized accurately. In this case, score following method[6,7] is very important.

By using such system, the player can concentrate upon the performance. However, the user sometimes wants to turn over the page before the playing position reaches the last bar of the page, for example, the user has already memorized the last measure of the page, or the beginning measure of the next page has complex passages. In such case, our system provides the page turning at any time. Therefore a combination of our system and fully automatic page turning system will bring about a more flexible page turning system.

7. Conclusions

This paper presented a system of automatic page turning of musical scores by detecting player's nods. We implemented the system by OpenCV on Java using a method of efficient face tracking. The nod estimation was based on hidden Markov model, and the system worked properly in real time.

As the future work, we will implement extensions such as turning to a previous page, coping with repeat with page jumping (da capo, Dal Segno, et cetera), introducing a fully automatic page turning system, and so on. Moreover, we have planned implementing this system in mobile devices like smartphones and tablets. Then, we will examine the ability of the system by a large number of subjects in real music performance.

Acknowledgement: This study is supported in part by Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science, No. 26240025.

References

1) Automatic page turning system (in Japanese), http://inforent.dreamblog.jp/blog/78.html
2) Christopher M. Bishop: Pattern Recognition and Machine Learning, pp.610-634 (Springer-Verlag, New York, 2006)
3) Java, https://www.java.com/en/
4) OpenCV, http://opencv.org/
5) PDFRenderer, https://developer.android.com/reference/android/graphics/pdf/PdfRenderer.html
6) A. Arzt, G. Widmer, and S. Dixon: Automatic page turning for musicians via real-time machine listening, Proc. of ECAI2008, pp.241-245 (2008)
7) E. Nakamura, T. Nakamura, Y. Saito, N. Ono, and S. Sagayama: Outer-Product Hidden Markov Model and Polyphonic MIDI Score Following, Journal of New Music Research, vol.43, no.2, pp.183-201 (2014)