Tokyo Institute of Technology
School of Engineering
Bachelor Thesis

Real-Time Tennis Ball Speed Analysis System Based on Image Processing

Supervisor: Associate Prof. Kenji Kise

February, 2016

Submitter
Department Computer Science
Student ID 12_11696
Name Paniti Achararit

 Supervisor Approval Mark

 Department Chair Approval Mark
Tennis ball speed analysis systems play an important role in improving player performance, predicting match results, and preparing strategies. For measuring tennis ball speed, speed guns and smart sensors have been widely used. Although these devices can provide relatively accurate measurements, they are expensive. Alternatively, we can build speed analysis systems using real-time image processing on personal devices such as smartphones and tablet PCs. This is a cost-effective approach because of the popularity of smartphones and tablet PCs in society.

In this paper, I propose a tennis ball tracking and speed analysis method based on image processing for real-time tennis ball speed analysis systems. My proposed method uses the information of blobs in consecutive frames to track the tennis ball and predict position of the ball when the camera could not capture it. The speed analysis is calculated by linear motion equation $v = \frac{s}{t}$ where $s$ is approximated to a half distance of a tennis court and time $t$ is calculated from the relation between the number of captured frames and the camera FPS.

The proposed method is evaluated in a real tennis environment. In the experiment, both straightly and diagonally serves are demonstrated. I use two evaluation metrics: tracking miss rate and error value. The error value represents the difference between the results compared with Smart Tennis Sensor. Besides, I evaluate the real-time efficiency based on the overall computation time.

My evaluation results show that the computation time of the proposed method is below camera FPS, and thus can be considered to be real-time. The proposed method has a maximum tracking miss rate of 10%, which is considered to be the unavoidable noises occurring in the background. When the tracking succeeds, the proposed method has a maximum error value of about 6%. The reason is considered to be the approximate distance and the camera performance. I discuss several solutions for alleviating these problems.
Contents

Chapter 1  Introduction  1
  1.1  Objective  ......................................................... 1
  1.2  Paper Outline  ....................................................... 1

Chapter 2  Tennis Ball Detection  2
  2.1  Color-Based Object Detection  ..................................... 3
  2.2  Motion-Based Object Detection  .................................... 3

Chapter 3  Proposal: Tennis Ball Tracking and Speed Analysis  8
  3.1  Initial Frame and Tennis Ball Selection  .......................... 9
  3.2  Tennis Ball Tracing  .................................................. 10
  3.3  Tennis Ball Speed Analysis by Captured Frame Number ...... 13

Chapter 4  Evaluation and Discussion  16
  4.1  Evaluation on Accuracy  .............................................. 16
  4.2  Evaluation on Real-Time Performance  ............................ 19
  4.3  Discussion  ............................................................. 19

Chapter 5  Conclusion  24

Acknowledgements  25

Reference  26

Publication  28
Chapter 1

Introduction

1.1 Objective

Tennis ball speed analysis systems play an important role in improving player performance, predicting match result and preparing a strategy. The device uses the information of sound or light, such as a speed gun [1, 2], is used to analyze object speed. It is also used to analyze tennis ball speed in the tennis game [3, 4]. Sony Corporation also launches Smart Tennis Sensor [5]. It analyzes tennis ball speed using the vibration that passed through the racket. Although these devices can provide relatively accurate measurements, they are expensive.

On the other hand, embedded system devices have been developed so far that we can build speed analysis systems using real-time image processing on personal devices such as smartphones and tablet PCs. This is a cost-effective approach because of the popularity of smartphones and tablet PCs in society. Thus, the objective of this study is to provide a low-cost, accurate and easy-to-use real-time tennis speed analysis system based on image processing. The accuracy of the proposed program is aimed to be the same as that of Sony Smart Tennis Sensor. A laptop computer is firstly used to study the quality and accuracy of the algorithm. In the experiment, both straightly and diagonally serves are demonstrated. The result of the proposed system is compared with the result of Sony Smart Tennis Sensor. Besides, the real-time efficiency is evaluated based on the overall computation time.

1.2 Paper Outline

This paper structure is written as the following. Firstly, the information on how to detect the tennis ball is presented in chapter 2. The proposal of tennis ball tracking method and speed analysis method is presented in chapter 3. The evaluation and discussion on the proposed method is presented in chapter 4. Lastly, the summary of this paper is presented in chapter 5.
Chapter 2

Tennis Ball Detection

Object detection is an important part of image processing. Since each image and interested object have the uniqueness, the accuracy and the performance of the program depends on how the object is defined and how the detecting method is chosen. In this study, we assume that camera and serving position is fixed at a position as in Figure 2.1 and there is no moving object in the background while the video camera is capturing.

Figure 2.1: Scenery from the Camera
2.1 Color-Based Object Detection

Color-based object detection is one of the simple methods that uses the RGB pixel value to detect the interested object. As tennis ball has a unique color known as fluorescent yellow or in the official name optic yellow, color based object detection is considerable to be a method for detecting the tennis ball [6].

RGB images contain 8-bit value of red, green and blue colors in each pixel. These 3 color combinations can display up to 16,777,216 different colors to the monitor which is good enough to represent the object’s real color. Although fluorescent yellow has the RGB value in decimal as (204,255,0) [7], the unstable environment may cause a change in the tennis ball color. The examples of these environments are light or shadow that reflex on the tennis ball, and a fast moving tennis ball that blends with the background when it is captured by the camera. RGB value range expanding is one of the solution for this issue, but it may include more noise in the result. This method may appropriate to be use in a light-color-stable environment, but an environment in reality is uncontrollable. Hence, this method is still not an appropriate method.

2.2 Motion-Based Object Detection

Motion-based object detection uses the difference of the object position between frames to detect the interested object. This method needs only the difference between frame, so instead of using a large 24-bit RGB image, a 8-bit grayscale image is more appropriate to be used. Moving tennis ball has an obvious motion between its position in each frame, therefore motion-based object detection is a suitable to be a method for detecting the tennis ball.

Background subtraction or foreground detection is one of the motion-based object detection methods that uses the difference of the current frame and a background model for tracking interested object. Output image is a binary image which is created by comparing each pixel with a suitable threshold level. The method’s result depends on how a background model is used. A little change in background environment may dramatically increase noises.

Frame difference method is similar to the background subtraction method, but it adjusts to the environment better. It calculates on multiple frames and continuously updates the background model which fits with the environment for this study. Therefore frame difference method is used for object detection in this study.
2.2.1 Grayscale

Grayscale image is the image that contains only 8-bit value in each pixel. Since the data size of a grayscale image is smaller than that of an RGB image, the image taken from the video camera is usually converted to a grayscale system before the calculation to obtain a better calculation speed. To convert an RGB image to grayscale image, one of the methods is to use the average value of the RGB color pixel for a pixel in grayscale image. The formula can be written as (2.2.1).

\[
Y = \frac{(R + G + B)}{3} \tag{2.2.1}
\]

where \(Y\) is grayscale, \(R\) is red, \(G\) is green and \(B\) is blue pixel value. This method is very simple, but it gives the unnatural grayscale image when \(R + G + B\) has the same value while the value of each color itself is different in each pixel.

According to Radiocommunication Sector of International Telecommunication Union (ITU-R) [8], the appropriate grayscale for most of digital televisions’ standard is recommended as (2.2.2).

\[
Y = 0.299R + 0.587G + 0.114B \tag{2.2.2}
\]

When the original image is Figure 2.2a, the average value of RGB color method gives the unnatural grayscale image as Figure 2.2b while the ITU-R recommended method gives the natural grayscale image as Figure 2.2c. The latter method is not only uses the average value, but also weights the value of each color for human perception. Additionally, it is possible to avoid the unnatural output pixel. Hence, the ITU-R recommended method is used.

Many studies also include Gaussian smoothing in order to obtain a better smooth grayscale. Gaussian smoothing or Gaussian blur is one of the filters, which is used to reduce noises that occur from the camera itself or the environment. The mechanism
is to apply a Gaussian function to each pixel in the image. Let \( x \) be a row pixel of the image, \( y \) be a column pixel of the image, and \( \sigma \) is the standard deviation of the Gaussian distribution, Gaussian function can be written as (2.2.3).

\[
G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2 + y^2)}{2\sigma^2}}
\]  

(2.2.3)

In this study the Gaussian blur is not applied because the image quality is good enough to use for the calculation without an additional calculation cost in the system.

---

**Figure 2.3:** Frame Difference Method Flow
2.2.2 Frame Difference Method

Figure 2.3 shows the overall step for frame difference method. This method is often used for tracking moving objects with a position fixed camera. Let $f_k$ be the $k^{th}$ frame, $f_l$ be the neighboring frame and $P(x, y)$ is a pixel of a row $x$ and column $y$ in an image. A function to find a foreground pixel $isFG_{f_k,f_l}(x, y)$ can be defined as

$$isFG_{f_k,f_l}(x, y) \overset{\text{def}}{=} (P_{f_k}(x, y) - P_{f_l}(x, y) > \delta)$$  \hspace{1cm} (2.2.4)

where $\delta$ is the threshold parameter defined by user. The result of the function is a binary image. The information of foreground pixel from each neighbored frame is then performed by a logical AND with the function $isFG_{f_k}(x, y)$ which can be defined as

$$isFG_{f_k}(x, y) \overset{\text{def}}{=} \bigcap_{l \in L} isFG_{f_k,f_l}(x, y)$$  \hspace{1cm} (2.2.5)

where $\cap$ is AND operation and $L$ is a set of neighbored frames. In this study neighboring frames $\{k-6, k-4, k+4, k+6\}$ are chosen to be the range of $L$. The reason why neighboring frames $k-2, k+2$ are not chosen because slow moving ball may deleted from the foreground image when the ball in neighboring frame is too close to that in the current frame [9].

2.2.3 Blob Detection

Binary large object (blob) detection is a function which is used to group connecting pixels in binary image into an object and then return the properties of it. It is also used to filter which blob should be detected or ignored by size, circularity, convexity or inertia ratio of the blob. Each of the filters can be defined by its maximum or minimum value.

In this study, blob detection function from OpenCV 3.0 are used to make a calculation in this step. According to camera perspective in Figure 2.1, tennis ball is considerable to be a small size blob. Therefore the binary image output from frame difference method is filtered by blob size filter. To get an adjustment for each environment, user is allowed to define the maximum and minimum blob size filter. Then each selected blob’s coordinate, size, and frame number is stored in an array. These informations can be used for labeling each blob as in Figure 2.4 and for analyzing speed as described in the following chapter.
Figure 2.4: Blob Detection Result
Chapter 3

Proposal: Tennis Ball Tracking and Speed Analysis

From the selected blobs in 90 consecutive frame images, object that is most likely to be a tennis ball will be traced and used for speed analysis. In this chapter the method for tennis ball tracking and speed analysis are proposed.

Figure 3.1: Defined Area
3.1 Initial Frame and Tennis Ball Selection

In order to trace a tennis ball, an initial frame and a tennis ball in that frame is selected and used as a base for tracing.

To track the ball easily, the captured image is divided into three areas as Figure 3.1: player area, detection area and net area. Blobs in the detection area can be considered to be a tennis ball because it is far away from noises that occurred from player movement and also the noise from the net. To get the frame that has the condition as above, function $containsBall(f_k)$ is used. The function checks each blob in the frame by the x,y coordinates.
If the center of a blob in the frame is in the detection area, which is satisfied when the blob position \((x, y)\) satisfies \((x > 0.3 \times \text{width}) \& (x < 0.8 \times \text{width}) \& (y > 0.8 \times \text{height})\), the function returns \(true\). In the above condition, the origin of row and column is at the top-left of the image, and \(\text{width}\) and \(\text{height}\) means the width and height of the image.

With this function, the frame that has blobs in the middle of the image is selected, but the blob can be either tennis ball or noise. To prevent noise selecting, 3 consecutive frames are used for checking the present of the tennis ball. If function \(\text{containsBall}(f_k)\) returns \(true\) for \(k = i, i - 1, i - 2\) and \(x_i > x_{i-1} > x_{i-2}\), then frame \(f_i\) is selected to be the initial frame and the satisfied blob is selected for tracing as in Figure 3.2.

### 3.2 Tennis Ball Tracing

After the first tennis ball candidate is selected, blob that is most likely to be a tennis ball in the previous frame and the next frame is traced. The approach is inspired by the trajectory generation by X. Zhou [10]. Tracing starts from the initial tennis ball to the player area for backward tracing and from initial tennis ball to net area in forward tracing.

![Figure 3.3: Blob Selection](image-url)
3.2.1 Backward Tracing

In backward tracing, tennis ball in the previous frame should stay on the left of the current one, therefore only blobs on the left side of the current one is considered. Then to prevent the unnecessary blobs from calculated, the region of interest (ROI) is applied by setting a maximum and minimum of tennis ball moving range. Then the nearest blobs from the current tennis ball is used for tracing in the second previous frame as Figure 3.3a. If there is no blob in the ROI, predicted position is used instead as Figure 3.3b. Predicted position is calculated by the (3.2.1) formula

\[
\text{Predicted Position} = \text{Current Position} + \text{Moving Average} \quad (3.2.1)
\]

when MovingAverage is calculated by the selected blobs position from initial frame to current calculation state. The operation is repeated until the latest selected blob position locates near the player area as Figure 3.4. The algorithm for backward tracing function can simply be written as below.

![Figure 3.4: Backward Tennis Ball Tracing Step](image-url)
1. search tennis ball in frame $f_{i-1}$
2. IF tennis ball is found THEN go to 4.
3. tennis ball = predicted ball
4. IF tennis ball is in player area THEN finish backward tracing
5. set $f_i$ to $f_{i-1}$ and go to 1.

3.2.2 Forward Tracing

Forward tracing is the opposite version of the backward tracing step. With the same idea, but only blobs on the right side of the current one is considered. The operation is repeated until the latest selected blob position locates near the net area as Figure 3.5. Then each selected position is saved to the output array. The
algorithm for forward tracing function can simply be written as below.

---

### Forward Tracing Algorithm

1. search tennis ball in frame $f_{i+1}$
2. IF tennis ball is found THEN go to 4.
3. tennis ball = predicted ball
4. IF tennis ball is in net area THEN finish forward tracing
5. set $f_i$ to $f_{i+1}$ and go to 1.

---

### 3.3 Tennis Ball Speed Analysis by Captured Frame Number

From the example data that executed by the method written in section 3.2, frame number 45 is chosen as the initial frame and the trace result is shown as Figure 3.6. After we know the information of the tennis ball in each frame, average serving speed is simply calculated by using linear motion formula $v = \frac{s}{t}$ while $v$ is the average ball speed, $s$ is the distance that tennis ball has moved and $t$ is traveled time.

In this paper, distance from serving area to tennis net is used, which can approximate to half of a tennis court. A tennis court is 78 feet (23.774 meters) long, so the distance $s$ is equal to 39 feet (11.887 meters).

Traveled time can be calculated from the number of frames that is divided by the camera frame rate (FPS). The number of frames can be taken from the tracing result as Figure 3.6. Therefore $t$ can be calculated by (3.3.1) formula.

$$t = \frac{number \ of \ frames}{FPS} [s] \quad (3.3.1)$$

Then, the complete formula for average serving speed is

$$v = \frac{11.887}{number \ of \ frames} \times FPS \ [m/s] \quad (3.3.2)$$

Using (3.3.2) formula, the speed result for Figure 3.6 is calculated as (3.3.3) and (3.3.4)

$$speed_m = \frac{11.887}{16} \times 30 = 22.2881 \ [m/s] \quad (3.3.3)$$
Table 3.1: The Change in Speed According to The Number of Frames

<table>
<thead>
<tr>
<th>Number of Frames (frames)</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tennis Ball Speed (km/h)</td>
<td>128</td>
<td>117</td>
<td>107</td>
<td>99</td>
<td>92</td>
<td>86</td>
<td>80</td>
<td>76</td>
<td>71</td>
<td>68</td>
<td>64</td>
</tr>
</tbody>
</table>

\[
speed_{km} = speed_m \times \frac{18}{5} = 80.2372 \text{ [km/h]} \quad (3.3.4)
\]

However, with the (3.3.2) formula, the speed result is limited to the same number as the frame number. When the tennis ball speed is between 60 to 130 km/h on the 30 FPS camera, the result is fixed as Table 3.1.

To get a better result, the number of frames need to be adjusted. With the relation between frame and tennis ball moving distance, the number of frames can be adjusted. By calculating a moving distance per frame, the start frame and the end frame are adjusted by the distance between its tennis ball and the defined area border as Figure 3.7. Let \( f_s \) be the first frame of the traced result and \( x_s \) be the
row pixel value of the first tennis ball. Start frame can be adjusted by the (3.3.5) formula.

\[
\text{adjusted start frame} = f_s + \left( \frac{1}{x_{s+1} - x_s} \times (\text{player area border} - x_s) \right) \quad (3.3.5)
\]

On the other hand, let \( f_e \) be the last frame of the traced result and \( x_e \) be the row pixel value of the last tennis ball. End frame can be adjusted by the (3.3.6) formula.

\[
\text{adjusted end frame} = f_e + \left( \frac{1}{x_e - x_{e-1}} \times (\text{net area border} - x_e) \right) \quad (3.3.6)
\]

Therefore, the adjusted number of frames is calculated from the (3.3.7) formula

\[
\text{adjusted number of frames} = \text{adjusted end frame} - \text{adjusted start frame} \quad (3.3.7)
\]

\[\text{Figure 3.7: Frame Adjustment}\]
Chapter 4

Evaluation and Discussion

4.1 Evaluation on Accuracy

In the experiment, Logitech video camera C920 is used to capture an image with 30 FPS and image processing is done with OpenCV 3.0 on Intel i7-2630QM 2.00GHz processor of laptop computer. Threshold level in frame difference step is set to 15. Minimum blob size and maximum blob size in blob detection step are set to 10 and 2000. The accuracy of proposed method is measured by comparing the proposed method’s result with Sony Smart Tennis Sensor’s results. The result is obtained from straightly and diagonally serving motion. Straightly serve is a serve that the tennis ball is parallel to the court line and diagonally serve is a normal serve according to the tennis rules.

Figure 4.1 and Figure 4.2 show the comparison results of the 30 experiments on straight serves and diagonal serves. Each mark represents the comparison result from the demonstration. y-axis shows the result of the proposed method and x-axis shows the result of Sony Smart Tennis sensor. The absent marks represent the miss traced result or the result that has an error value above 20%. The straight blue line $y = x$ shows how many results are faster or slower than Sony Smart Tennis Sensor’s results and the difference between the results.

From Figure 4.1,4.2 and Table 4.1, the result has no significant trend in straightly serve, while the number of the slower result in diagonally serve is 1.875 times greater.

<table>
<thead>
<tr>
<th></th>
<th>Straightly Serve</th>
<th>Diagonally Serve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample (times)</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Absolute Error (km/h)</td>
<td>4.37</td>
<td>3.88</td>
</tr>
<tr>
<td>Percent Error (%)</td>
<td>5.94</td>
<td>5.17</td>
</tr>
<tr>
<td>Miss Rate (%)</td>
<td>3.33</td>
<td>10</td>
</tr>
</tbody>
</table>
Figure 4.1: Straightly Serving Result

It has 4.37 km/h absolute error equal to 5.94% error in straightly serve and 3.88 km/h absolute error equal to 5.17% in diagonally serve. There is no any result that has an error value above 20%.

During the experiment there are unavoidable noises occurring in the background which cause 3.33% miss rate in straightly serve and 10% in diagonally serve. The causes of error are considered to be the approximated distance, camera performance limit and time which is approximated by the number of frames.
Figure 4.2: Diagonally Serving Result
4.2 Evaluation on Real-Time Performance

Figure 4.3 shows the concept of how computation time of the program is measured. \( t_s \) is the time when the camera starts capturing frames. \( t_c \) is the time when 90 consecutive frames and the method in chapter 2 is completely computed. \( t_e \) is the time when the program end. State 1 represents the time used for capturing and preparing the image, which can be calculated as \( State1 = t_c - t_s \). State 2 represents the time used for tracking the tennis ball and analyzing its speed, which can be calculated as \( State2 = t_e - t_c \).

In 10 measurement the proposed program gives the result as Table 4.2. It uses 2.9987 seconds in State 1 and use 0.0006 seconds in state 2. The total time used for computing the program is 2.9993 seconds. Since the 30 FPS camera use 3 seconds to capture 90 frames and the computation time does not exceed the camera FPS, the proposed program is considered to be real-time.

4.3 Discussion

There are many reasons that lead to a miss tracing and the inaccurate result. In this section the problem and the solution are discussed.

Table 4.2: Computation Time Result

<table>
<thead>
<tr>
<th>Sample (times)</th>
<th>Proposed Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Time Used in State 1 (s)</td>
<td>2.9987054</td>
</tr>
<tr>
<td>Time Used in State 2 (s)</td>
<td>0.0006136</td>
</tr>
<tr>
<td>Total Time (s)</td>
<td>2.999319</td>
</tr>
</tbody>
</table>
4.3.1 Causes of Miss Tracking

The reason of miss tracking is the unavoidable noise from the environment as mentioned above. In initial frame selection step, noises which appear in the detection area for a while before the initial frame is selected, will cause the false initial frame selection and lead to the miss tracking. Moreover, in tennis ball tracking step, noises near the tennis ball may also cause the miss tracking.

To deal with these problems, noise reduction function is one of the considerable options. However, the tennis ball pixel in the image is not so different from noise pixel, so the function possibly deletes the tennis ball and may lead to another problem. Another considerable option is to re-select the blob for tracing or re-select the initial frame by rolling the program back. When a blob locates in the prohibit area while tracing, the program is rolled back and another blob is selected to trace. The prohibit area is defined as the area when the center of a predicted ball is below or exceed frame height. Then, if the program is rolled back to the initial frame and there is no other blob to select, a new initial frame is selected. The new algorithm for backward and forward tracing can be written as below.

--- Tennis Ball Tracking with Roll Back Algorithm ---

1. search tennis ball in frame \( f_{i-1} / f_{i+1} \)
2. IF tennis ball is found THEN go to 4.
3. tennis ball = predicted ball
4. IF tennis ball is in player position THEN finish backward / forward tracing
5. IF tennis ball is in prohibit area THEN roll back
6. set \( f_i \) to \( f_{i-1} / f_{i+1} \) and go to 1.

4.3.2 Causes of Error Value

Approximated Distance

The proposed method uses a constant distance for the formula \( v = s / t \), which is one of the reasons of the error value. The distance in a straight line is used while the serve is not always straight. This causes a slower trend in the result of diagonally serve as in Figure 4.2.

In a normal serve, tennis ball should moving across the court diagonally. The approximate distance based on the diagonal line should be newly defined as Figure 4.4.
Although the new distance is defined, the error value still remains because the ball moving distance is not a constant value. The additional function for approximating the distance is a considerable option. From the camera perspective as Figure 2.1, the diagonally moving ball path has its characteristic. The gap between the ball in each frame tends to increase as it moves closer to the camera. The wider the gap is, the more angle the tennis ball is served. By using the ball characteristic to approximate the distance, a better result is expected to be obtained.

**False Prediction**

Another reason is the predicted value used when the camera could not track the tennis ball. As mentioned in section 3.2, predicted value uses the tennis ball moving average. With the average value and the condition to use only blob on the left in backward tracing step, a predict position might locate beyond the real position as Figure 4.5. After the false prediction occurs, the predicted position is continually used and cause an error result. By weighting more value to the moving distance that is close to the current frame, the more accurately predicted position is considered to be obtained.

**Camera FPS**

In the proposed method, the camera FPS determines the traveled time of the ball. Assume that the camera in the experiment is a 60 FPS camera, the tennis ball speed
precision is simply 2 times greater as shown in Table 4.3. With the higher FPS, the more accurate result is obtained.

The high FPS camera helps to increase the accuracy, but the time per frame also has an influence on the result. The stable 30 FPS camera has an ideal time per frame at 0.033 second. With the stable camera, the traveled time of the tennis ball is accurately captured. On the other hand, unstable camera influence the time and cause an error as in Figure 4.6. The error of the time not only affect the start and end frame, but also affect on the moving distance per frame which lead to a more error when approximating tennis ball position.

To know the error value caused by unstable camera, it is simply measured by recording a time after capturing a frame and finding the difference between the current time and the previous time. Table 4.4 shows the result of Logitech C920’s unstable time per frame. The result shows that the camera has 1.67 milliseconds error in each frame which equal to 5%. According to the linear formula \( v = s/t \), \( t \) is a direct variation to \( v \). Since in the proposed method camera FPS is used as \( t \),

Table 4.3: The Change in Speed According to The Number of Frames Between 30 and 60 FPS Camera

<table>
<thead>
<tr>
<th>Tennis Ball Speed (km/h)</th>
<th>107</th>
<th>103</th>
<th>99</th>
<th>95</th>
<th>92</th>
<th>88</th>
<th>86</th>
<th>83</th>
<th>80</th>
<th>78</th>
<th>76</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 FPS Camera (frames)</td>
<td>12</td>
<td>-</td>
<td>13</td>
<td>-</td>
<td>14</td>
<td>-</td>
<td>15</td>
<td>-</td>
<td>16</td>
<td>-</td>
<td>17</td>
</tr>
<tr>
<td>60 FPS Camera (frames)</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
</tr>
</tbody>
</table>
the error from the camera itself is unavoidable. Therefore the error in the result is considered to have this 5% error from the camera included.

**Table 4.4**: Logitech C920 Web Camera SPF Compared with ideal SPF Result

<table>
<thead>
<tr>
<th>Sample(frame)</th>
<th>Absolute Error(ms)</th>
<th>Percent Error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logitech C920 SPF</td>
<td>120</td>
<td>1.67</td>
</tr>
</tbody>
</table>
Chapter 5

Conclusion

In this paper, tennis ball tracking and speed analysis method based on image processing for real-time tennis ball speed analysis systems is proposed. It is evaluated by tracking miss rate and error value on 30 straightly serve and diagonally serve result.

As the result, the computation time of the proposed method is much less than the capturing time, and thus can be considered to be real-time. There is about 5-6% error in the result. The main reasons are considered to be the approximate distance and the camera performance limitation. Since there is 5% error from the unstable camera, the error value can be considered to be low. There is a maximum miss rate of 10%, which occurs from unavoidable noises in the background. According to these results, the proposed method is low-cost and the analyzed speed is not so different from Sony Smart Tennis Sensor.

In order to obtain better results, tennis ball tracking with roll back algorithm and the approximate distance value need to be further investigated. The easy-to-use application for iOS tablet PCs is also expected to be implemented in future work.
Acknowledgements

Firstly, I am thankful to God for the good health and good mentality that is needed to complete this paper.

The completion of this paper could not have been possible without the support of the following people. My supervisor, associate professor Kenji Kise, he is the one who encourages me to do this project, allows the paper to be my own work and gives the right support whenever I need it.

I would like to specially thank to Mr. Yuki Matsuda for helping me along with the project. He always teaches me a new knowledge, gives me the new idea and perspective through the problem, and help me carry out the experiment. Without his guidance and passionate participation, the project would not successfully conducted.

I would like to sincerely thank Mr. Susumu Mashimo for providing devices for the experiment. He allows me to use his laptop PC and new web camera. He also encourages me.

I would also like to acknowledge Mr. Masashi Fujinami for his discussion and valuable comments on the project and his support throughout these 4 years at the university.

I would also like to express gratitude to all members of the Kise Laboratory for their help and support.

I am thankful to Thai Government for the great opportunity and the scholarship. This project would have been impossible without the support.

I also place on record, my sense of gratitude to all, who directly or indirectly, lend their hand in this venture.

Finally, I am deeply grateful to my parents and every Thai friends for providing me a good environment and encouragement throughout my year of study. This accomplishment would not have been possible without them. Thank you.
References


http://www.sony.co.jp/SonyInfo/design/works/products/tennis\_sensor/01.html.


[7] colorhexa, “#ccff00 color information,”
http://www.colorhexa.com/ccff00.


Publication

Oral Presentation