Seeing while walking

Daning Yang
Li Zeng
Yao Wang
Yu Cao
Prof. Alexander Medvedev

Department of Information Technology
Uppsala University
2006-05-28
(Latest version 2006-08-28)
Abstract

This report describes a solution to stabilize the axis of the camera on AIBO, a robot from SONY. First, comprehensive understanding of the AIBO functional process is obtained from various testing. Second, an information acquisition method is defined to measure the movement of the AIBO’s head while it is walking. Third, by introducing a program written in Matlab, an offline module is found to compensate for the unwanted head movement based on data collected in the previous step. Finally, the AIBO uses that offline module to predicate the movement of the head, which is done by synchronizing with the walking style. The reason causing the unwanted head movement is analyzed, which is the background knowledge to develop a stabilization algorithm. This report also describes the limitations of the solution and suggestions for further research on how to stabilizing AIBO’s head while walking.
Abstract.......................................................................................................................... 2
1. Introduction .................................................................................................................. 4
   1.1. Background ............................................................................................................. 4
   1.2. Hardware Information .......................................................................................... 4
   1.3. Software information ............................................................................................ 6
   1.4. Previous Work ................................................................................................ ...... 8
       1.4.1. Physical modeling ......................................................................................... 8
       1.4.2. Head leveler .................................................................................................. 8
       1.4.3. Camera movement measurement .................................................................... 8
2. AIBO Head Shaking Measurement ............................................................................. 9
   2.1. Terms Definition ................................................................................................... 9
   2.2. Image Sequence .................................................................................................. 10
   2.3. Camera Shaking and Head Movement ............................................................... 10
   2.4. Object in the Image Sequence ............................................................................. 10
   2.5. Object Center Movement Detection Algorithm .............................................. 12
   2.6. Camera Shaking Measurement ............................................................................ 13
3. Walking Style and Camera Shaking .......................................................................... 15
   3.1. Background knowledge ....................................................................................... 15
   3.2. WalkMC .............................................................................................................. 15
   3.3. Motion Locus and Camera shaking ...................................................................... 16
4. Compensation Control Curve ..................................................................................... 18
   4.1. General ................................................................................................................ 18
   4.2. Offline Model Building ....................................................................................... 18
       4.2.1. Camera Shaking Curve Adjustment .............................................................. 18
       4.2.2. Compensation Function ............................................................................... 19
       4.2.3. Indexed Compensation Curve ...................................................................... 19
   4.3. Online Movement Control .................................................................................. 20
5. Experiments and Results ............................................................................................ 21
6. Conclusion .................................................................................................................... 23
1. Introduction

1.1. Background

AIBO is the robot dog manufactured by SONY with the first model coming into the market in 1999, which later presented in the auto robot football championship RoboCup. This project actually came from a practical problem encountered by the engineers, who try to make AIBO react to the picture stream taken by its camera on the head. For various reasons the AIBO's camera presents fiercely shaking pictures, which makes it impossible to recognize objects of human interest, such as the goal, the goal keeper and the opponent player, etc, and the timely and exact recognition of these objects is essential to the success of the robot football game. Therefore there must be a solution to reduce the camera shaking to an acceptable level to make the detecting and tracking of an object possible, and further measuring velocity of an object, which requires a sequence of even more stable images. There are a number of algorithms dealing with this kind of problems based on image processing. However image processing is too much a heavy task for the AIBO’s limited hardware configuration. As tentative experiments illustrated, if the CPU is occupied by image processing task implemented for camera shaking correction, AIBO can not walk smoothly anymore, say nothing of other actions like kicking the ball. A spastic football player is definitely not helpful on the ground.

This leaves few choices, which is later described and discussed in this report. An offline model is constructed and applied to the online movement of AIBO, by adjusting the AIBO’s head movement to stabilizing the camera, and consequently reducing the shaking of the objects in the image stream.

1.2. Hardware Information

There are several different AIBO models in the market. The one revolved in this project is ESR-210, of which the hardware information is shown below in Table 1 and 2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera sensor</td>
<td>100 000 Pixels CMOS</td>
</tr>
<tr>
<td>Camera</td>
<td>176x144 pixel images @ 25 fps in YUV color</td>
</tr>
<tr>
<td>CPU</td>
<td>64-bit RISC processor</td>
</tr>
<tr>
<td>Memory</td>
<td>32Mb</td>
</tr>
<tr>
<td>Removable flash memory</td>
<td>8MB “Memory Stick for AIBO programming”</td>
</tr>
<tr>
<td>Power</td>
<td>rechargeable 2300mAh lithium battery</td>
</tr>
</tbody>
</table>
### Table 2 Movable Parts and Movement Freedom

<table>
<thead>
<tr>
<th>Name</th>
<th>Freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chin</td>
<td>1 degree</td>
</tr>
<tr>
<td>Neck</td>
<td>3 degrees</td>
</tr>
<tr>
<td>Leg x 4</td>
<td>3 degrees of freedom x 4</td>
</tr>
<tr>
<td>Ear x 2</td>
<td>1 degree x 2</td>
</tr>
<tr>
<td>Tail</td>
<td>2 degrees</td>
</tr>
</tbody>
</table>

The optional limits (freedom) are illustrated in Figure 1 and 2.

**Figure 1: Operational Limits of Head**

**Figure 2: Operational Limits of Legs**

**Figure 3 Software Operational Limits of Head**
Inside the AIBO body, there is also an acceleration sensor (CPC Primitive Locator) which could sense the movement acceleration in x, y, and z direction.

1.3. Software information

SONY created a platform on which programmers could develop in C++ language, namely OPEN-R\(^1\). In fact, this is the lowest level down to which SONY allowed the customers to reach. However, OPEN-R provides the maximum freedom known until now to control AIBO grounding on complex installation and coding\(^2\).

Another friendlier platform for AIBO is Tekkotsu\(^3\). This is a kind of wrapper of OPEN-R platform. On Tekkotsu platform, programs in C++ language are uploaded into the memory stick and work under OPEN-R. The AIBO side codes are very well encapsulated. Lots of classical C++ idea, like templates, smart pointers, are presented in this design and the codes are elegant (This can be great material in an advanced C++ design course). Tekkotsu platform provides a well designed remote control user interface in JAVA, which works on UNIX system via wireless LAN. The UI gives an illuminating idea of how AIBO can walk, shout, hear, and see in the easiest and fastest way. Moreover, since there are pre-coded monitors and controllers, Tekkotsu is also a good example to learn how to work with AIBO. Put it simple, it is easier to start with Tekkotsu, also with enough freedom. See Figure 4.

![Figure 4 Tekkotsu Main Controller](image)

In Tekkotsu system, file ERS2xxInfo.h includes definitions of great information and limitations for developing program to control AIBO, such as joints number of each leg, maximum output speed, variable storing sensor’s output etc. These software limitations defined by Tekkotsu may vary from hardware limitations introduced by SONY. For instance,

\(^1\) More information can be found on http://openr.aibo.com
\(^2\) There is a group working on OPEN-R platform in spring, 2006. Their report might be include more information with regard to OPEN-R
\(^3\) More information can be found on http://www.tekkotsu.org
software output range of head joints is a little smaller than the mechanical limitation mentioned above. Before starting program under Tekkotsu framework, it should always be a good idea to read file ERS2xxInfo.h that enables the users to access the up-to-date actual state and ability of each AIBO.

Tekkotsu provides a platform, on which environment parameters configurations of vision, motion, sound etc. could be done conveniently. Please refer to file ‘tekkotsu.cfg’ to get detailed information of user setup. Each line of configuration in the file is well commented, which makes it quite straightforward to understand. Here, several important parameters that are closely related to image stream capturing deserve some further explanation.

- ‘white_balance’, ‘gain’ and ‘shutter speed’
  Adjusting any of these three parameters, pixel color value will change. It influences, usually, the threshold value in a certain image processing algorithm if pixel color is used as a standard to, for instance, find edge or identify object. Yet any combination of these parameters will not increase CPU’s burden.

- ‘resolution’
  This parameter is quite obvious. A full resolution corresponds to camera hardware limitation, 176*144 pixels, from which half and quarter resolution can be derived. If the complexity of an online algorithm scales with the number of pixels in one image, full resolution requires CPU time the most.

- ‘rawcam_encoding’, ‘rawcam_compression’, ‘rawcam_compress_quality’
  These parameters affect how images taken by raw camera algorithm are presented and then saved either in memory or file. For example, by setting rawcam_encoding, camera saves grey images instead of colorful ones. If user saves the images on Tekkotsu GUI side, grey images are smaller than colorful ones, which obviously shorten the time needed to transfer them via network. Correspondingly, parameters are available for rle camera and region camera

- ‘walk’, by which user selects walking style\(^1\).

All mentioned parameters are loaded by class Config\(^2\).

Finally, here is a tip for debugging: stdout in the program running on AIBO’s side is redirected to telnet port 59000 by default. Please note that, unlike outputting characters to console, redirecting stdout via network is not a synchronized operation, which means if program crashes while running, it is not guaranteed that all output data buffered in network cache will be sent out. So, for the following piece of code:

```c
printf("Crash!!!\n");
int *p = NULL; *p = 0;
```

More often than not, ‘Crash!!!’ will not appear on the telnet console.

\(^1\) Through our this report, walking style is always referring to Tekkotsu’s ‘walk.prm’

\(^2\) See ‘config.h’ for more information
1.4. Previous Work

1.4.1. Physical modeling

This is a method \[^{3}\] which tried to calculate the position of the camera that relative to gestures of AIBO. For example, since detailed parameters of external appearance of AIBO is available, it is possible to know camera’s position by some geometric calculation on the ground of the angles of each leg joints, the leg’s length, body length and width, and etc.. However, there is a flaw in this model, because part of a walking period, at least one leg is in the air. The situation is even worse if using Tekkotsu’s walking style\[^{1}\], because AIBO moves two legs, either front-left and back-right or front-right and back-left, synchronously. With only two legs touching the ground, this method is infeasible for most of the time.

1.4.2. Head leveler

Head leveler seems to be a ‘terrific idea’\[^{3}\], which used accelerometers, independent of AIBO’s body placement, to regulate neck joints to return to the wanted state, to result a stabilized camera. Unfortunately, experiment did not show a good result as expected. Accelerometers are too sensitive. Curve produced according to accelerometer’s output data changes so frequently and sharply that exceeds the limited response time of head’s joints.

1.4.3. Camera movement measurement

In the experiment of group 2004, a big dot was used as the object toward which AIBO walked. In the experiment of group 2005\[^{2}\], a black cross printed on white paper was used instead and the motion track of the cross was measured to compute the headshaking. The method of this project is an enhanced version of their original idea, and is explained in the following chapters.

\[^{1}\] More information can be found in section ‘Walking Style and Camera Shaking’
2. AIBO Head Shaking Measurement

2.1. Terms Definition

Before presenting the measurement, several terms should be defined to describe the head's movement.

- **Tilt**: rotation about X-axis
- **Pan**: rotation about Y-axis
- **Roll**: rotation about Z-axis

Finally, in this report the following descriptors are also used.

- **H(α, β, δ)**
  Head position descriptor in which ‘α’ stands for head tilt degrees about X-axis, ‘β’ for pan degrees about Y-axis, and ‘δ’ for roll degrees about Z-axis.

- **W(f, s, r)**
  Walking velocity descriptor in which ‘f’ stands for forward, the velocity on Z-axis, ‘s’ for strafe, the velocity on X-axis, and ‘r’ for rotate, the angular velocity about Y-axis.
2.2. Image Sequence

By using Tekkotsu’s monitor tool, one can save image sequence taken by the camera placed on AIBO’s head in PNG format.

‘Save Image Sequence’ on ‘Vision Raw’ GUI would transfer image sequences from the AIBO to the host computer in PNG format and save them in the same folder along with an automatically generated images index file in txt format. Both the names of the folder directory and the index file are user defined. The names of the PNG files consist of a prefix of the name of the index file and a suffix of a ‘time stamp’ which is the time, in millisecond, elapsed since the first image is saved. Following is an example:

   Index file name: new.txt
   An image file name: new000919.png

2.3. Camera Shaking and Head Movement

As mentioned before AIBO only has one camera, what AIBO sees is only a 2-D image sequence. Therefore, the camera shaking problem could be considered as only related to the AIBO’S headshaking in the X-Y plane. Therefore, if a 2-D object is placed in front of AIBO and parallel to the X-Y plane, the movement of the object along the image sequence could indicate the camera shaking.

2.4. Object in the Image Sequence

To produce enough scale for AIBO’s low camera resolution (176x144 pixels), a 2-D object, which is referred to as the Cross in the following part of the report, consists of two black decussated lines, which both have 80-pixels in width, and spreads on 9 pieces of A4 paper that are connected together.

In order to reduce the CPU load, the camera resolution is reduced to half of its maximum value. In this case, AIBO’s camera is able to take usable photos at about one and a half meters away from the Cross.

The Cross is placed against the wall under sufficient light resource (fluorescent light is adequate and direct sun light is proved as excessive). AIBO’s head position and walking speed should be adjusted and specified before a walk starts, and these steps are easy to be carried out with the help of Tekkotsu’s ‘Walk Remote Control’ and ‘Head Pointer Control’ panels. Figure 7 to Figure 9 are acquired from internet resource[^4]

[^4]: Tekkotsu generates file name is automatically
Figure 7 Walk Remote Controller

Figure 8 Head Pointer Control

Figure 9 Camera Controller
2.5. Object Center Movement Detection Algorithm

Edge detection is applied to the raw PNG image file. Since the lines of the Cross are 80 pixels, the edge image is also cross shaped as shown below in Figure 10.

![Figure 10 Edge of The Cross](image)

![Figure 11 Center Pixel](image)

The center pixel index (CP_Inx) of the Cross in this image is then calculated as illustrated in Figure 11.

After all the PNG images listed in the index file are processed, a sequence of center pixel indices is acquired. The center movement of the \(i\)th image \(CM(x_i, y_i)\) is computed according to the following equation.

\[
CM(x_i, y_i) = CP_{Inx}(x_i, y_i) - CP_{Inx}(x_0, y_0)
\]

In which \(CP_{Inx}(x_0, y_0)\) stands for the center pixel position in the first PNG file. The movement of the Cross center between two pictures is then equal to the center indices difference.

The algorithm is shown below.

```plaintext
Function begin
    Read index file
    Do
        :: img = next PNG file according to index
        Edge(img)
        X = center movement on X-axis
        Y = center movement on Y-axis
        Save [X Y]
    End of index -> break
    [code]
End
```
2.6. Camera Shaking Measurement

Because the object is standing still, the calculated object movement is caused by camera shaking only. Thus, the measurement of camera shaking is the same in the value as the object movement measurement.

A sample movement of the Cross is plotted according to the x and y index value, respectively, as shown below in Figure 12 and 13, in which X-axis stands for the index of the center pixel and Y-axis for movement value in pixels.

It should be noticed that the movement before center pixel number 20\textsuperscript{th} is comparatively small since the first step of an AIBO walk is softer than the following steps. Further experiment illustrated that the center movement curves are similar in shape, on X and Y axis, respectively. Therefore, it could be concluded that the camera shaking has its general patterns.

It should be also noticed that both the movement on X and Y axis could be considered as roughly periodic, if we consider a turning point of the curve in the lower part is the start position, and a turning point of the curve in the upper part is the end position of the period, as shown in Figure 14.
Thus the camera shaking amplitude of one step could be measured by calculate the absolute difference between the movement value of the start and end point, since they are the minimum and maximum, respectively of their corresponding period. Similarly, the camera shaking amplitude of one whole walk could be measured by calculate the mean value of all the steps it contains.
3. Walking Style and Camera Shaking

3.1. Background knowledge

Tekkotsu separates source files into two categories by their functions, which are controller behavior (the WalkCB), and walking motion control (the WalkMC). The WalkCB acquires velocity values in three directions (tilt, pan, rotation) from user via wireless LAN. The WalkMC namely control the walking behavior of the AIBO.

3.2. WalkMC

Theoretically, given a start point, an end point, and the in-between points, a curve could be produced as a ‘spline’. An example is shown in Figure 15.

As in AIBO walking behavior, there is only one in-between point, which is referred to as ‘the neutral point’. Understandably, if the start point A of a step as well as the step velocity is provided, the end point B of the step could be calculated. A function in WalkMC called ‘Spline’ could generate the step motion locus according to point A, point B, the neutral point X and the velocity. Figure 16 shows an example of a step motion locus of one leg, in which point A is where the leg starts to lift, point B is where it hits the ground, and X is the neutral point. Small black dots other than A and B on the locus is the imaginary point where WalkMC is invoked by ‘MotionManager’, which is scheduled by Tekkotsu, so as to update leg’s joints value.

The motion locus could be defined as \( p = F(t) \), in which ‘p’ is the leg’s space position and \( t \) the percentage of the period. The WalkMC generates the \( F(t) \) of the next period for a leg.
when it is touching the ground. During one walking period, if the AIBO’s velocity is not zero, ‘Motion Manager’ would invoke WalkMC periodically and the leg joint angular value is calculated according to $F(t)$ following kinematics laws.

In real case, WalkMC imports a walking style parameter file into a data structure called ‘WalkParam’ as it is initialized, for leg motion locus calculation. In this project, the file name is ‘walk.prm’.

```c
struct LegParam {
    vector3d neutral; // defines the "neutral" point of each leg - where it is in midstep
    vector3d lift_vel; // give the velocities to use when raising the paw
    vector3d down_vel; // give the velocities to use when lowering the paw
    double lift_time; // the time (as percentage of WalkParam::period) in the cycle to lift
    double down_time; // the time (as percentage of WalkParam::period) in the cycle to put down
};

struct WalkParam {
    LegParam leg[4]; // a set of LegParam's, one for each leg
    double body_height; // the height to hold the body (mm)
    double body_angle; // the angle to hold the body (rad - 0 is level)
    double hop; // sinusoidal hop amplitude
    double sway; // sinusoidal sway in y direction
    long period; // the time between steps
    long useDiffDrive; //non-zero, diff-drive turning is used instead of rotational turning
    float sag; // the amount to sagging to account for when a foot is lifted
    float reserved; // just live with it
};
```

The joints of the AIBO’s leg adjust according to the angular value to achieve the position in the motion locus. Specifically, to simulate a natural walk, the left-front and right-back is defined as one pair of simultaneously moving legs, and the other two legs the other pair, respectively. Assuming that the backlash is very small, the camera shaking is guessed to be caused largely by the inclination when there are only two legs supporting AIBO’s moving body and also the shock coming from legs when they are hitting the ground.

### 3.3. Motion Locus and Camera shaking

To prove the guess mentioned in previous section, the camera shaking and leg motion locus are compared to each other with the percentage of the walking period as the uniform reference, as shown in Figure 17. The first sub plot is camera shaking curve in X direction and, the second is Y direction camera shaking, and the third is the legs’ motion locus. The object movement does not precisely repeat because of the unpredictable factors like loose head joints that cause backlash or unbalanced leg motion that causes velocity deviation. But the object movement in X and Y direction has a fluctuant trend corresponding to legs motion locus in all periods.
**X direction:** The curve rises when left leg lifts up, and turns when left leg steps down. Accordingly, a drop occurs when right leg lifts up and turns when right leg steps down. It could be deduced specifically that in the beginning of one period of AIBO's walk, AIBO lifts up left-front(LF) and right-back(RB) legs while its right-front(RF) and left-back(LB) legs is touching the ground. In the same time, AIBO’s body starts to incline toward left-front because of gravity until the LF leg hits the ground, and a camera shaking peak value is produced as shown in the curve named ‘Scaled Object Movement in X Direction’. When LF and RB legs keep unbending, RF and LB legs starts to lift up, then put down and finally hit ground, which is reflected as a trough.

Therefore the cross movement has a “sine curve” shape.

**Y direction:** Since movement in Y direction is perpendicular to the walking route, the leg of different sides has the same influence on the movement. The curve rises when leg lifts up and returns when leg steps down. Therefore there are two rises in one walking period.

Figure 17 Comparison of Object Movement in X, Y dir and Leg’s Motion Locus
A conclusion could be made that the camera shaking is periodic and synchronized to the walking behavior. And therefore, it is possible to adjust neck joints purposefully, synchronizing to percentage of walking period, to compensate for the camera shaking.
4. Compensation Control Curve

4.1. General

If the camera’s position moves from the initial point $P(x_0, y_0, z_0)$ to point $P(x_i, y_i, z_i)$, the shaking value is defined as $S_i = P(x_i, y_i, z_i) - P(x_0, y_0, z_0)$. Theoretically, change the head position by $-S_i$ would fully compensate for the shaking. Similarly, the camera shaking compensation during walk could be achieved by feeding different $S_i$ at each point. But the real case, many complicate factors should be taken account of. In this project, an offline model contains actual compensation value $C_i$ is generated and applied to the AIBO.

4.2. Offline Model Building

4.2.1. Camera Shaking Curve Adjustment

As shown in Figure 12 and Figure 13 in chapter 2.6, the camera shaking curve value has a trend, because that the object moves from the center of the camera to a upper area along walking. To detrend the curve, the camera shaking of each walking period is normalized by the mean camera shaking of the period.

$$A_{\text{detrended}}(i) = A_{\text{original}}(i) - \frac{1}{n-m} \sum_{k=m}^{k=n+1} A_{\text{original}}(k)$$

In which $m$ is the index of the first point of a period, and $n$ is the index of the last point. But still the camera shaking amplitude of each period is steadily increasing, because that the scale of the object increased when the distance between the object and the camera decreased.

Figure 18 Detrend Camera Shaking Curve in X and Y Direction
4.2.2. Compensation Function

First it was considered to derive two high-order finite functions for compensation in X and Y direction, in which percentage of period is the input and neck joints value is the output. To strike a balance between approximation accuracy and AIBO control performance, the two functions have power of 8 in X direction and power of 10 in Y direction, respectively. But this is still a heavy burden for CPU and AIBO moves in jerk sometimes. This is because that CPU is consumed by WalkMC for too much time during its calculation of value of neck joint. CPU and memory is usually contradictory. Therefore, a method that utilizes more memory and less CPU time is introduced.

4.2.3. Indexed Compensation Curve

This method aims to feed compensation value at 16 control points according to the average camera shaking curve. Since a period would either contain 16 or 17 sampling points, the average camera shaking is only computed from those periods of 16 sampling points. See Figure 19 for an example.

![Figure 19 Original Index Compensation Curve](image)

The above curve should not be applied as the indexed compensation curve because of the backlash effect. This problem is actually an important reason of the shaking. In fact, any newly introduced control on the neck joint brings new shakings. However, this disadvantageous influence could be take advantage of by reducing difference between the compensation values of two adjacent points. In other words, the indexed compensation curve only compensates for part of the shaking and the rest of the work is done by the backlash caused by the compensation control.

Moreover, the control and the effect could not be simultaneous, because there is a delay before neck joints are updated. The delay is introduced by task schedule manager and should be constant if no new operation is assigned to CPU. Therefore, further adjustment in both value and phase to the compensation curve is necessary. This was done empirically after experiments aiming to find the best compensation result.

\(^5\) A difference smaller than 2 is acceptable by experience
4.3. Online Movement Control

Indexed compensation curves for X direction and Y direction have 32 parameters in total, which are stored in file ‘Param.txt’ under the root directory of AIBO’s Memory Stick. These values are loaded into memory along with file ‘Walk.prm’ and used to generate and setup neck joints compensation value as leg joints are updated. Here is a piece of code:

OffsetOfPeriod = (int)(cycle * 16); // cycle is the percentage of period

double t=m_fAY * fPT[OffsetOfPeriod]; // compensation value in X direction.
double p=m_fAX * fPP[OffsetOfPeriod]; // compensation value in Y direction.

Read ‘NewWalkMC.cc’ for detail information.
5. Experiments and Results

Figure 20 is a comparison of camera shaking curve (black cross) and compensated camera shaking curve (red dot) in X direction and Y direction. Shaking amplitude is obviously reduced.

![Figure 20 Camera Shaking Curve and Compensated Camera Shaking Curve](image)

Table 3 is a summary of ten times of camera shaking mean and ten times of compensated shaking mean in X direction.

### Table 3 Summary of Camera Shaking in X Direction

<table>
<thead>
<tr>
<th>Shaking Mean</th>
<th>10.55</th>
<th>10.32</th>
<th>13.90</th>
<th>12.03</th>
<th>11.28</th>
<th>10.04</th>
<th>10.23</th>
<th>10.23</th>
<th>10.91</th>
<th>11.31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compensated Shaking Mean</td>
<td>4.50</td>
<td>4.00</td>
<td>6.00</td>
<td>4.79</td>
<td>5.31</td>
<td>5.17</td>
<td>4.95</td>
<td>4.95</td>
<td>5.25</td>
<td>5.28</td>
</tr>
</tbody>
</table>
Table 4 is a summary of ten times of camera shaking mean and ten times of compensated shaking mean in Y direction.

<table>
<thead>
<tr>
<th>Shaking Mean</th>
<th>Compensated Shaking Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.66</td>
<td>8.00</td>
</tr>
<tr>
<td>16.17</td>
<td>8.28</td>
</tr>
<tr>
<td>18.31</td>
<td>9.89</td>
</tr>
<tr>
<td>14.90</td>
<td>7.18</td>
</tr>
<tr>
<td>18.78</td>
<td>10.06</td>
</tr>
<tr>
<td>21.59</td>
<td>10.64</td>
</tr>
<tr>
<td>20.01</td>
<td>10.06</td>
</tr>
<tr>
<td>20.01</td>
<td>10.17</td>
</tr>
<tr>
<td>19.04</td>
<td>11.33</td>
</tr>
<tr>
<td>20.99</td>
<td></td>
</tr>
</tbody>
</table>

Table 4 Summary of Camera Shaking in X Direction
6. Conclusion

This solution could be universal to all kinds of AIBO's walking style. For different style, one might simply use our toolkits to generate the indexed compensation curve, then edit 'param.txt' file we used to store these compensation value. However, several factors influence positive effectiveness of camera compensation as follows.

1. Walking style and field on which AIBO is walking. It is easy to understand that different style call for different indexed curve, while various fields influence camera pattern greatly. That AIBO walks on a hard wood ground makes camera greater than it walking on soft carpet. However the effect of backlash is less if AIBO walks on a softer ground, which needs a more smooth indexed compensation curve. If influence of backlash, the unpredictable factor, becomes smaller, the positive effect of compensation by indexed compensation curve is bigger.

2. Indexed compensation curve reflects a relationship between object movement (unit: pixel) in picture taken by AIBO's camera and percentage of period of walking. To compensate for the unwanted object movement in picture, we adjust AIBO's head joints to 'put' the object back to where it is supposed to be. So, there must be a function to convert pixel that object moves to angle that the head should turn. By trigonometry, it is clear that the distance from AIBO's camera to the inspected object is a key factor. Indexed compensation curve should be scaled according to how far AIBO is looking at. Obviously, scale increases as AIBO walking towards the object and decrease as it leaving the object. A proper scale algorithm would make this solution more practical.

3. More indices of indexed compensation curve might produce better result. The image sequence used to produce indexed compensation curve is taken by camera in maximum 17 frames. Because either wireless LAN process or CPU is almost fully consumed, no more images are taken within a second. However, 'MotionManger' invokes the WalkMC far more than 17 times per second, which means value of leg joints are updated more frequently. We can double the number of indices from 16 to, say, 32, so that head joints are adjusted more precisely. This increases memory usage only and do not add a little burden to CPU.

Moreover, translate movement on X and Y axis has a maximum amount of 5 pixels from the original center. This measurement is definitely not very precise. Due to the low resolution of AIBO's cameras, the images are blurred and the edges presented are not the real edges of the object and after segmentation the “false edges” could further deviate from the original positions. Miscalculating of one pixel can leads to 10% error. Therefore it is possible that some very small fluctuations on the curve are not real camera's motion, but belongs to miscalculating. This is a problem that this report does not solve but left for future work.

With respect to the trend in these two directions, the movement deduced from the image sequence is processed using a “detrend” algorithm to maximally reduce the negative influence. Admittedly, this detrend does not elaborate the influence completely. However,
the relationship between cross position movement and the period of AIBO legs movement can be found which is expected for the reason that AIBO's walking style is something periodically repeating and AIBO's motion directly cause.

Furthermore, in the same walking style the AIBO could move at different speed, which results in different movement behaviors. To make it more complicate, as mentioned before, each AIBO has own performance parameters in uniform walking style, and a RoboCup player should definitely be content to move in different styles. Therefore, in practical use, a constant offline model building is not sufficient. Since it is impossible to put an online task on AIBO to generate a model, an alternative is to build several offline models for each AIBO and choose the appropriate one according to the current walking speed and style. To achieve more accurate control, before contest, the human backup team could run the program and calibrate AIBO to obtain new control models.

In summary, this solution is sound, because it does not occupy too much CPU time and produces an acceptable result of reduction of camera. It might be possible to embed a training program into AIBO so that it can produce indexed compensation curve in some ways. This gives AIBO an ability to adapt to different ground and saves technician's calibration time for different AIBO.

References

[1] Model Information for ESR-210
[2] Seeing while walking, report from 2005