Semester Project
Summer 2007

Programming The Robotic Dog AIBO

Pierre-Arnaud GUYOT

Supervisor: Sarah DEGALLIER, BIRG EPFL
Responsible Professor: Auke Jan IJSPEERT, BIRG EPFL

School of Computer & Communication Sciences
Swiss Federal Institute of Technology Lausanne (EPFL)

June 29 2007

Contents

List of Figures

1 Introduction

1.1 Main objectives

The objective of my semester project is to program the AIBO ERS7 robotic dog. Created by Sony in 1999, AIBO is a commercially available quadruped robot equipped with a color CMOS camera. The aim of my work is to make AIBO able to detect a mark on the floor using his camera. Then it will place his nearest paw on the detected mark.

In this work, there are three main objectives. The first one is to develop a robust mark detection algorithm by using image processing techniques. The second objective is to create a 2D to 3D mapping from the position in the picture to the position in the floor. Finally the third objective is to develop an inverse kinematics based leg controller. The entry of this controller is the previously computed position of the mark in AIBO reference frame, and the outputs are the leg joint angles that make the paw reach the mark. A first simulation of the algorithms will be done using the mobile robot simulator Webots. Then the program will be implemented in the real dog by using cross compilation.

1.2 Analysis of the different solutions

1.2.1 Program organization

The program can be separated in 3 different parts:

- First a mark detection algorithm detects the mark in the pictures. This algorithm returns the 2D position of the mark in the picture.
- Then a 2D to 3D mapping function converts the 2D picture position into a 3D position relative to AIBO.
- Finally a controller calculates the joints angles values and activate the 3 servos of one leg to make the AIBO's paw reach the mark.

All the tasks are implemented in a finite state machine represented in figure 1.

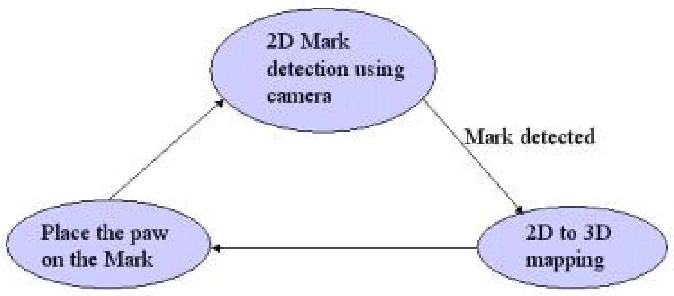


Figure 1: Program organization

1.2.2 Mark detection

Two solutions are possible to detect the mark in the picture:

- Color recognition: By knowing in advance the color of the mark, we can compute its position in the picture by using a virtual mask moving on all pixels: The mean color inside the mask is calculated to find the mark position in the picture.
- Shape recognition: Image processing algorithms allows us to find circles or lines in the picture. In the image processing course of Mr. Jourlin [1], we can see that circle recognition needs first different filters applied on the picture to detect the edges (Roberts, Prewitt or Sobel filters). After extracting edges, circles can be detected by using shape parameters (Crofton Parameters) and lines are detected by using Hough transform.

For the 4 following reasons, we will adopt the color recognition method:

- The camera perspective distortion modify the shape of the mark and could disturb the shape recognition.
- The shape recognition algorithms need more computational power due to the image filtering and edge extraction.
- The floor's color will be very different from the mark color, the color recognition technique can then be efficient.
- Color recognition algorithms are easier to implement than shape recognition.

1.2.3 2D to 3D mapping

We need to convert the mark position in the picture into a position in the AIBO coordinates system. In order to establish the correspondence *2D position in the picture / 3D position in the AIBO coordinates system*, 3 solutions are possible:

- We can use the AIBO ER7 simulation in Webots to construct 2D to 3D correspondence matrices. These matrices of size $pictureheight \cdot picturewidth$ contains the positions on the floor corresponding to each pixels of the picture. To construct these matrices, a supervisor makes move a very small square on the floor in front of the AIBO camera such that each pixels of the picture 'see' successively the small moving mark (this mark is seen as 1 to 2 pixels by the camera depending on the perspective). A supervisor in Webots does the link *pixel(i,j)/ 3D position on the floor*. For each pixel (i,j), the corresponding 3D mark position is placed on the position (i,j) in the matrix. Finally to get the 3D position corresponding to a pixel (i,j), we need only to read the value (i,j) of the matrix.

  Note that the real AIBO camera and the modeled camera in Webots have the same field of view and resolution, we can then use such matrices constructed in Webots with the real robot.

- A mathematical solution can be use to map from 2D to 3D. This method called camera calibration consists of calculating the matrices of the geometrical transforms that leads a 2D point in the picture into a 3D point in the AIBO coordinates system.
- The 3D position of the mark can be found by triangulation. At least two images should be taken at two different positions.

The first solution has yet been implemented to test the program, this solution is adapted only if we know in advance the positions of the AIBO's body when the pictures are taken (one matrix per body position is needed). The calibration method is adapted when we don't know in advance in which direction the camera will be oriented. The calibration matrix must be computed for each different body position. The third solution appears to be not adapted for monocular vision.

1.2.4 Placing the leg on the mark

The 2D to 3D mapping gives us the 3D position of the mark in AIBO coordinates system. The legs joint values are then computed by using inverse kinematics.

1.2.5 Simulation and implementation on the real robot

The simulation in Webots allows us to validate the models used to calculate the calibration matrix and to compute the joints values. The image processing algorithms can also be tested in Webots. To implement the algorithms on the real robot, 3 solutions are possible:

- Cross-compilation of the Webots program: the Webots controller code is cross-compiled to produce a binary executable which then directly runs on the robot . In this case, the binary executable must just be copied on the AIBO memory stick.
- Remote-control mode: A standalone computer program enables controlling and monitoring of Aibo over a wireless network connection. The controller runs as part of Webots simulation engine (from cyberbotics [8]).
- OpenR programming: The whole program is directly written in OpenR language and placed on the AIBO memory stick.

The remote-control mode does not yet support camera. To use the remote control solution, Webots should be modified to be able to get the images that AIBO sends to the computer.

The cross-compilation does not support functions related to the camera. The cross compiler is written in OpenR language and can be easily modified to compile functions related to the camera. Re-witting manually the Webots in OpenR language is quite long, that's why the cross-compilation method has been adopted.

2 Mark detection

2.1 Mark color and shape

The color of the mark to be detected should be different from the floor and environment colors to facilitate the detection, that's why a red mark will be used. The simulation in webots shows that a simple red square can be easily detected.

2.2 Mark detection using virtual mask

A virtual mask goes through all pixels of the picture. The total amount of red inside the mask is computed for each mask position. The 2D position of the mark corresponds to the mask position where the amount of red is maximum. To avoid detecting light sources as mark, the amount of blue and green should be lower than a threshold. In the same time, the amount of red should be higher than an other threshold to be sure that the detected object is the mark.

Note: The values of these thresholds are determined by doing experiments with the real AIBO in his environment.

3 Mapping from 2D to 3D

The 2D position of the mark in the picture is now computed by the mark detection algorithm. We will now describe the calibration method which permits to compute the 3D position of the mark. For the next sections, we will use homogeneous coordinates. Inversions or combinations of linear transformations are then represented by inversion or multiplication of the corresponding matrices (Roger P. Woods, course on homogeneous coordinates[8]). Instead of representing each point (x,y,z) with a single three-dimensional vector, we extend the coordinates to 4 dimension by adding ω to x, y and z:

$$\begin{bmatrix} \omega \cdot x \\ \omega \cdot y \\ \omega \cdot z \\ \omega \end{bmatrix}$$

3.1 Camera transform: 3D to 2D

We should now establish the correspondence *2D position in the picture / 3D position relative to AIBO*. To find a matrix representing this correspondence, let´s think about the reversed problem: What are the 2D coordinates (in the picture) of a point expressed in the AIBO reference frame? We want to find a matrix A such that:

$$\begin{bmatrix} w_i \cdot x_i \\ w_i \cdot y_i \\ w_i \cdot z_i \\ w_i \end{bmatrix} = A \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

With:

- (x, y, z) : Coordinates of the point in the AIBO coordinates system.
- (x_i, y_i, z_i) :Coordinates of the point in the plan of projection ($z_i = const.$).

Note that the point $(x', y') = \{0, 0\}$ corresponds to the center of the picture, $(x', y') = \{$-1, -1$\}$ corresponds to the upper right corner of the picture.

The transformations that lead a point in the AIBO coordinates system to a pixel in the picture are explained in A. Rovetto and F. Scandelli thesis [4] and illustrated in figure 2. First the *World transform* leads the AIBO framework to the camera framework. Then the perspective projection consists of projecting the 3D point into the 2D image plane. The picture in the image plane is finally sampled into pixels.

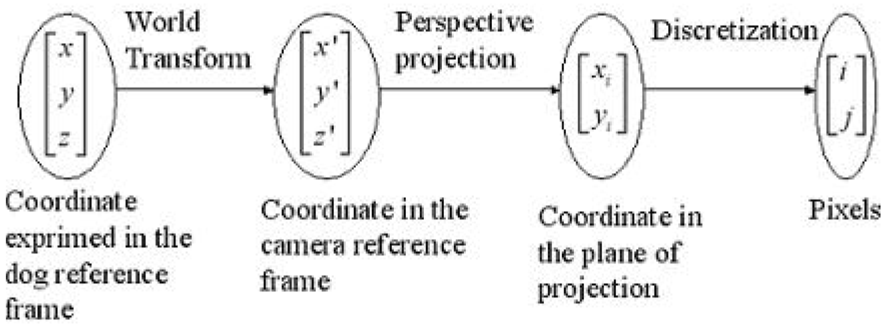

World Transform
Perspective projection
Discretization
x y z
x' y' z'
x_i y_i
i j
Coordinate exprimed in the dog reference frame
Coordinate in the camera reference frame
Coordinate in the plane of projection
Pixels

Figure 2: Transforms to obtain the pixel corresponding to a 3D point

3.1.1 *world transform*

For the sections 3, we will use the following conventions and notations:

- $ROTx/y(\theta)$ means a rotation around the x/y-axis of an angle θ.
- $Trans \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$ means a translation of the vector $\vec{T}(t_x, t_y, t_z)$.
- The origin of the AIBO framework is placed on the tilt center (see figure 3.b), $\vec{x}$ is pointing in front of AIBO and $\vec{y}$ points on the left.

First, the point expressed in the AIBO coordinates system should be written in the camera coordinates system. This is achieved by using one translation of vector $\vec{v}(t_x, t_y, t_z)$ and 3 rotation around the 3 axis $\vec{x}, \vec{y}, \vec{z}$ (from Wikipedia, 3D projection[6]).

$$M = Trans \begin{bmatrix} a \\ b \\ c \end{bmatrix} \cdot Rot_X(\alpha) \cdot Rot_Y(\beta) \cdot Rot_Z(\gamma)$$

The relation is equivalent to:

$$M = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot$$

$$\begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & -\cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

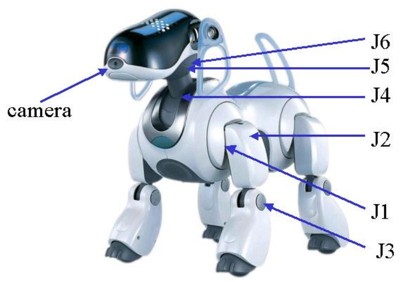
Matrices multiplication results in

$$\begin{pmatrix} \cos(\beta)\cos(\gamma) & -\sin(\alpha)\cos(\beta) & \sin(\beta) & a \\ \sin(\alpha)\cos(\beta)\cos(\gamma) + \cos(\alpha)\sin(\gamma) & -\sin(\alpha)\cos(\beta)\sin(\gamma) - \cos(\alpha)\cos(\gamma) & -\sin(\alpha)\cos(\beta) & b \\ -\cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) & \sin(\beta)\cos(\alpha)\sin(\gamma) - \cos(\gamma)\sin(\alpha) & \cos(\alpha)\cos(\beta) & c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

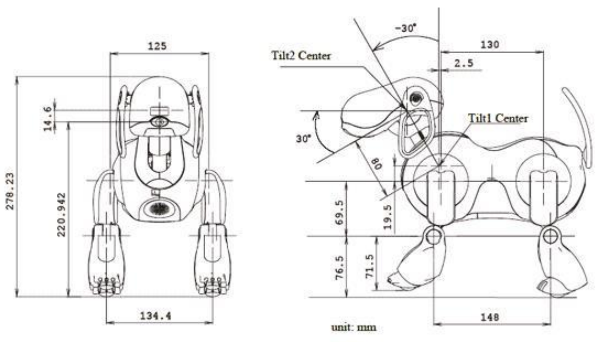
The six parameters $a, b, c, \alpha, \beta, \gamma$ corresponds to the camera position and orientation in the AIBO framework. To determine these values in Webots, we make the AIBO reference frame coincide with the Webots World reference frame, after what we read in webots the fields *translation* and *rotation* corresponding to the camera framework.

With the real AIBO, we will need to find the matrix M by using direct kinematics. AIBO is modeled by a chain of joints separated by links. M represents the chain of transforms that leads the AIBO framework to the Camera framework. According to the 3 degrees of freedom J4,J5 and J6 (see figure 3.a), the transforms are the following:

- A counterclockwise rotation around the $\vec{y}$ axis of angle θ_4 due to the joint J4 (Tilt Neck): $ROT_y(\theta_4)$
- A translation of h_1 along $\vec{z}$-axis: $Trans \begin{bmatrix} 0 \\ 0 \\ h_1 \end{bmatrix}$
- A counterclockwise rotation around the $\vec{y}$ axis of angle θ_5 due to the joint J5 (Tilt Head): $ROT_y(\theta_5)$
- A counterclockwise rotation around the $\vec{z}$ axis of angle $\theta_6 + \pi/2$ due to the joint J6 (Pan): $ROT_z(\theta_6)$.



(a)



(b)

Figure 3: a)Tilt, neck and front left leg joints, b)ERS7 schematics from Webots[6]

- A translation of h_3 along the $\vec{z}$ axis: $Trans \begin{bmatrix} 0 \\ 0 \\ h_3 \end{bmatrix}$

We obtain the '*world transform*' matrix M by multiplying the last transforms matrices:

$$M = ROT_y(\theta_4) \cdot Trans \begin{bmatrix} 0 \\ 0 \\ h_1 \end{bmatrix} \cdot ROT_y(\theta_5) \cdot ROT_z(\theta_6) \cdot Trans \begin{bmatrix} 0 \\ 0 \\ h_3 \end{bmatrix}$$

3.1.2 Perspective transform

The 3D point which is now expressed in the camera coordinates system is then projected into the plane z=F (see figure 4).

The perspective projection matrix is the following (from Wikipedia, 3D pro-

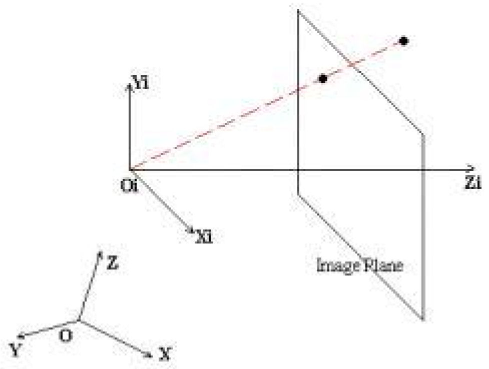


Figure 4: Perspective projection

jection[6]):

$$T = \begin{pmatrix} \cot(fov_x/2) & 0 & 0 & 0 \\ 0 & \cot(fov_y/2) & 0 & 0 \\ 0 & 0 & 1 & -2F \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{1}$$

With:

- fov_x : View port's horizontal fields of view, in radians.
- fov_y : View port's vertical fields of view, in radians.
- F : Distance of the observer from the front clipping plane.

We can now multiply M and T to find the final transform matrix called the calibration matrix. A point P(x, y, z) in the AIBO reference frame corresponds to the position P$(x_i/w_i, y_i/w_i, z_i/w_i)$in the picture such that:

$$\begin{pmatrix} x_i \\ y_i \\ z_i \\ w_i \end{pmatrix} = M \cdot T \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2}$$

Note: x_i and y_i are divided by w_i to obtain values in the range [-1; 1].

3.1.3 Sampling

The point P$(x_i/w_i, y_i/w_i)$ of the projection plan is integrated in a pixel (i, j) of the picture. If $(0, 0)$ corresponds to the pixel $(0, 0)$ we obtain:

$$x_i = 2.(i - i_0).w$$
$$y_i = 2.(j - j_0).h$$

With:

- w: Picture width in pixel.
- h: Picture height in pixel.

3.2 Camera Calibration

The 3 unknowns $fov_x,\ fov_y$ and F correspond to the intrinsic parameters of the camera. To find them, a calibration method can be used:

A very small mark is placed in front of the camera. By knowing the position P(x, y, z) of the mark in the AIBO coordinates system, the matrix M and the position P'$(x_i/w_i, y_i/w_i)$ of the mark in the picture, we can find the three unknowns $fov_x,\ fov_y$ and F by solving the system of equations (2).

3.3 Mark position in AIBO reference frame

After finding the two matrices M and T, we can establish the correspondence *position in the picture / position in the AIBO reference frame.* This mapping is done by computing the matrix $(M \cdot T)^{-1}$. The matrix M represents a bijection, M is then invertible. The matrix T is invertible if his determinant is not equal to 0. We can calculate $det(T)$ by doing linear combinations on the lines 3 and 4 of T:

The determinant of T is:

$$det \begin{pmatrix} \cot(fov_x/2) & 0 & 0 & 0 \\ 0 & \cot(fov_y/2) & 0 & 0 \\ 0 & 0 & 1 & -2F \\ 0 & 0 & 1 & 1 \end{pmatrix} = det \begin{pmatrix} \cot(fov_x/2) & 0 & 0 & 0 \\ 0 & \cot(fov_y/2) & 0 & 0 \\ 0 & 0 & 1 & -2F \\ 0 & 0 & 0 & 1-2F \end{pmatrix}$$

$$= det \begin{pmatrix} \cot(fov_x/2) & 0 & 0 & 0 \\ 0 & \cot(fov_y/2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \cot(fov_x/2) \cdot \cot(fov_y/2)$$

$\cot(fov_x/2) > 0$ and $\cot(fov_y/2) > 0$, T is then invertible, $M \cdot T$ is then invertible. For each pixel under the horizon line, the system of equation (2) has an unique solution.

4 Inverse kinematics

After calculating the desired foot position (position of the mark in the AIBO reference frame), the joint angles leading to this foot position are determined by using inverse kinematics method. In the following section, only the solution for the left front leg will be described. To get the joints angles for the right leg, the value of q_2 should be negated. The Robot Cup German team [2] has developed the following approach using first forward kinematics to get a set of equations, and then inverse kinematics to find one by one the joints values.

4.1 Forward Kinematics

The forward kinematics problem consists in the calculation of the resulting foot position for a given set of joint angles. Finding solution to this problem will allow us to solve the inverse kinematics problem. For this section, the origin of the framework corresponds to the shoulder. The figure 5 presents the left front leg of AIBO and the axis orientation.

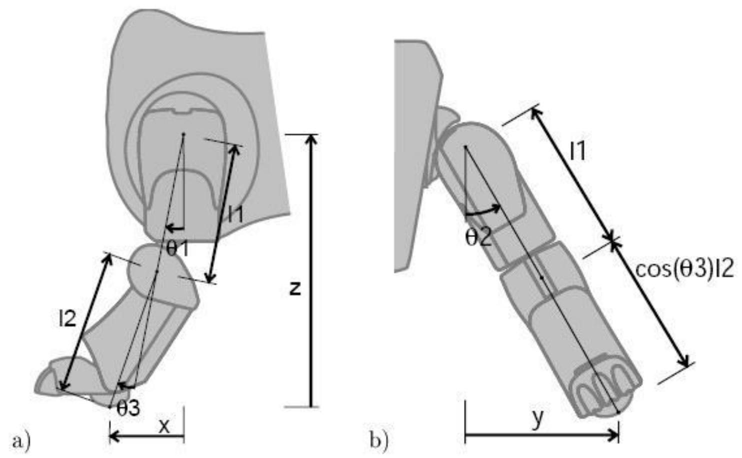


Figure 5: Left front leg with extents and angles. a)side elevation for the calculation of knee joint angle q3. b) front elevation for the calculation of shoulder joint angle q2 (taken from Uwe Duffert[5]).

The foot position relative to the shoulder can be calculated by using the composition of 5 transforms. The chain of transforms is the following:

- clockwise rotation about the y-axis by joint angle q_1 on joint J1: $ROT_y(-q_1)$
- counterclockwise rotation about the x-axis by joint angle q_2 on joint J2: $ROT_x(-q_2)$
- translation along the negative z-axis by upper limb length l_1: $Trans\begin{bmatrix} 0 \\ 0 \\ -l_1 \end{bmatrix}$
- clockwise rotation about the y-axis by joint angle q_3 on joint J3: $ROT_y(-q_3)$
- translation along the negative z-axis by lower limb l_2: $Trans\begin{bmatrix} 0 \\ 0 \\ -l_2 \end{bmatrix}$

To calculate the foot position relative to the shoulder, we need to multiply together each transformation matrices. With transforms expressed using homogeneous coordinates we obtain:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = ROT_y(-q_1)\cdot ROT_x(q_2)\cdot Trans\begin{bmatrix} 0 \\ 0 \\ -l_1 \end{bmatrix}\cdot ROT_y(-q_3)\cdot Trans\begin{bmatrix} 0 \\ 0 \\ -l_2 \end{bmatrix}\cdot\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{pmatrix} \cos(q_1) & 0 & -\sin(q_1) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(q_1) & 0 & \cos(q_1) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(q_2) & -\sin(q_2) & 0 \\ 0 & \sin(q_2) & \cos(q_2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(q_3) & 0 & -\sin(q_3) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(q_3) & 0 & \cos(q_3) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The relation between (x, y, z) and the angles q_1, q_2 and q_3 is then,

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} l_2\cos(q_1)\sin(q_3) + l_2\sin(q_1)\cos(q_2)\cos(q_3) + l_1\sin(q_1)\cos(q_2) \\ l_1\sin(q_2) + l_2\sin(q_2)\cos(q_3) \\ l_2\cos(q_1)\sin(q_3) - l_2\cos(q_1)\cos(q_2)\cos(q_3) + l_1\cos(q_1)\cos(q_2) \\ 1 \end{pmatrix} \tag{3}$$

4.2 Inverse kinematics

Inverse kinematics consists of finding the angles q_1, q_2 and q_3 from a given foot position (x, y, z). We will first determine q_3 with the law of cosine, after what we will use the relation (3) to find one by one q_2 and q_1.

4.2.1 Calculation of knee joint angle q_3

By fixing the knee joint angle q_3, we restrict the foot positions to a sphere around the shoulder. The angle q_3 can be calculated from the distance of the foot position (x, y, z) to the shoulder joint. Assuming that AIBO can reach the mark with his paw, we can use the law of cosines to calculate q_3. The angle included by thigh l_1 and lower leg l_2 is $\pi - q_3$ (see Figure 6). According to the law of cosines, we have:

$$\cos(\pi - q_3) = \frac{l_1^2 + l_2^2 - (x^2 + y^2 + z^2)}{2l_1l_2}$$

$$\cos(q_3) = \frac{(x^2 + y^2 + z^2) - l_1^2 + l_2^2}{2l_1l_2}$$

if $\frac{(x^2+y^2+z^2)-l_1^2+l_2^2}{2l_1l_2}) \in$[-1,1] then

$$q_3 = \pm\arccos(\frac{(x^2 + y^2 + z^2) - l_1^2 - l_2^2}{2l_1l_2})$$

Two solutions are possible. According to the the joint limitation (larger freedom of movement for positive q_3), we will chose the positive value:

$$q_3 = \arccos(\frac{(x^2+y^2+z^2) - l_1^2 - l_2^2}{2l_1l_2})$$

4.2.2 Calculation of knee joint angle q_2

The angle q_2 can be directly calculated according to equation (3):

$$y = l_1 \cdot \sin(q_2) + l_2 \cdot \sin(q_2) \cdot \cos(q_3)$$

$$y = sin(q_2) \cdot (l_1 + l_2 cos(q_3))$$

if $\frac{y}{l_1 + l_2 \cdot \cos(q_3)}$ ∈[-1,1] then

$$q_2 = \arcsin(\frac{y}{l_1 + l_2 \cdot \cos(q_3)}) \tag{4}$$

or

$$q_2 = \pi - \arcsin(\frac{y}{l_1 + l_2 \cdot \cos(q_3)}) \tag{5}$$

Due to the joint angles limitations, we will use the relation (4) to calculate the angle q_2.

4.2.3 Calculation of knee joint angle q_1

According to the equation (3):

$$x = l_2 \cdot \cos(q_1) \cdot \sin(q_3) + l_2 \cdot \sin(q_1) \cdot \cos(q_2) \cdot \cos(q_3) + l_1 \cdot \sin(q_1) \cdot \cos(q_2) \tag{6}$$

As q_2 and q_3 are calculated, the equation (5) can be simplified by introducing the variables a, b, c, d, β (see figure 6):

$$a = l_2 \sin(q_3)$$

$$b = (l_1 + l_2 \cdot \cos(q_3)) \cdot \cos(q_2)$$

$$d = \sqrt{a^2 + b^2}$$

$$\beta = \arctan(\frac{b}{a})$$

a and b can also be described in polar coordinates by using d and β

$$a = d \cdot \cos(\beta)$$

$$b = d \cdot \sin(\beta)$$

By introducing these variables in the equations (6) we obtain:

$$x = a \cdot \cos(q_1) + b \cdot \sin(q_1)$$

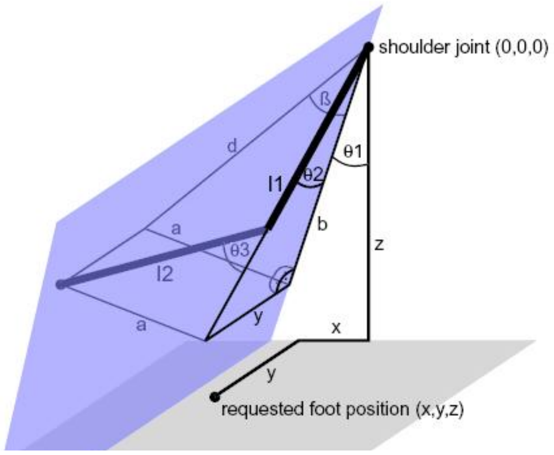


Figure 6: Model of AIBO leg for the calculation of shoulder joint angle q_1 with all auxiliary variables (from Uwe Duffert[5]).

$$x = d \cdot \cos(q_1) \cdot \cos(\beta) + d \cdot \sin(q_1) \cdot \sin(\beta)$$

$$x = d \cdot \cos(q_1 - \beta)$$

and with analog substitutions:

$$z = d \cdot \sin(q_1 - \beta)$$

If x $\neq$0 we obtain

$$q_1 - \beta = \arctan(\frac{z}{x})$$

$$q_1 = \arctan(\frac{z}{x}) + \beta$$

To summarize, we obtain the following joint angles:

$$q_3 = \arccos(\frac{(x^2 + y^2 + z^2) - l_1^2 - l_2^2}{2 l_1 l_2})$$

$$q_2 = \arcsin(\frac{y}{l_1 + l_2 \cdot \cos(q_3)})$$

$$q_1 = \arctan(\frac{z}{x}) + \arctan(\frac{(l_1 + l_2 \cdot \cos(q_3)) \cdot \cos(q_2)}{l_2 \cdot \sin(q_3)})$$

4.2.4 Anatomy correction

By sending a 0 degree consign to all the joints, we obtain two angles q_{10} and q_{20} on the joints q_1 and q_2 (see figure 7). We should then add the constants (-q_{10}) and q_{20} respectively to q_1 and q_2 when we fix the real joints values:

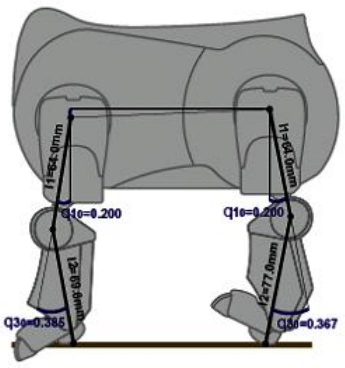


Figure 7: Position of the joints when we send an angle of 0 degrees as consign (from Uwe Duffert[5])

$$q_1 := q_1 - q_{10}$$
$$q_2 := q_2 - q_{20}$$

5 Program simulation and implementation

5.1 Webots simulation

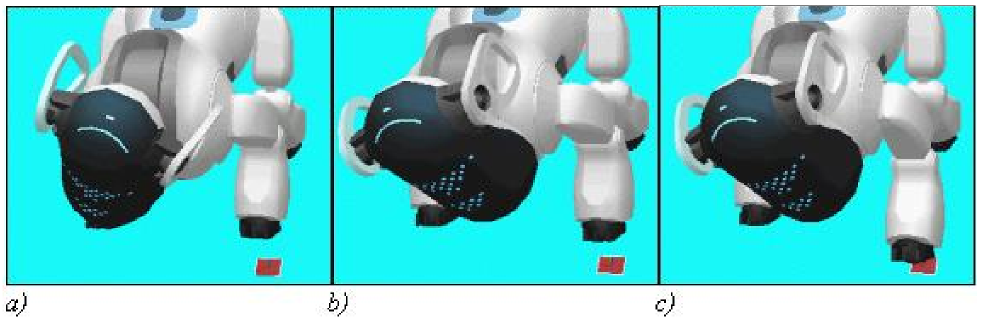


Figure 8: Program simulation in Webots: a) AIBO search the mark on left and right direction, b) and c) AIBO reach the mark.

The model of the AIBO ERS7 takes into account all servos and the camera. The field of view and the resolution of the webots's camera are the same that the real one. The webot's camera provides images formatted in the RGB color spaces. In the real AIBO camera, the pixels are coded in the YUV color space.

The Webots simulation is very useful in this project for the following reasons:

- The three main components (see figure 1) of the program can be tested.
- The 2D to 3D mapping can be elaborated (see section 1.2.3). The 2D to 3D correspondence matrices can be easily constructed by using a supervisor in Webots. If we need a more robust 2D to 3D mapping, a camera calibration matrix can also be computed. As we have seen previously, this calibration matrix is constructed with the combination of the *world transform* matrix and the perspective projection matrix. The Webots simulation enable to check if these matrix are correctly calculated.
- The program of the simulation written in C language can be cross-compiled to be directly transfered into the real AIBO.

The Webots simulation enable to test the robustness of the mark detection algorithm by creating environments of various colors and lighting.

In the figure 8, we see AIBO searching the mark and placing his paw on the mark. The webots program is divided in 4:

- Initialization: All joints values are initialized.
- Mark detection: AIBO grab a picture. If there is a mark, the mark detection function returns the pixel corresponding to the center of the mark.

- 2D to 3D: Find the 3D position corresponding to a pixel.
- Inverse kinematics: Compute the joints values to reach the 3D position of the mark.

5.2 Program cross-compilation

Softwares for the AIBO are written in Sony's object-oriented language called OPEN-R. A cross-compiler written in OpenR allows us to re-use the Webots simulation program. The idea of cross-compilation is to translate the Webots functions into OpenR language. First the Webots program is compiled with OpenR SDK cross-compiler, the binary files are then transfered into the AIBO memory stick. By rebooting AIBO, the binary files are directly executed. Today, the cross-compiler does not compile the functions related to the camera such that camera_get_image() or camera_image_get_red(). To do a cross-compilation of the whole Webots program, the five following functions must be added into the cross-compiler:

- One Camera_get_image() function to grab a picture from the camera.
- 1 function to convert the picture from the YUV to the RGB color space.
- 3 functions to get the R, G or B values of a pixel.

The function that converts YUV to RGB color space has been implemented in OpenR language during the Raphaël Haberer-Proust semester project [3]. This function is included in the remote control program for Aibo (RCServeur). I'm currently implementing the functions Camera_get_image() and camera_image_get_red().

By using cross-compilation to implement the program in OpenR, remote control is not possible. The whole program will then run on the AIBO´s 576MHz processor which is powerful enough to do fast image processing and motion control tasks.

6 Conclusion

The Webots simulation of the program is efficient. In a first time, the modeled AIBO detects the mark on the floor, then 2D to 3D correspondence matrices are used to determine the 3D mark position and the leg joints values are computed through inverse kinematics. To have a more robust program, the 2D to 3D mapping could be done by computing the calibration matrices described in the section 3.

To enable AIBO to achieve more complex tasks, some modules could be added to the current program: When the mark is not reachable, a walk module that makes the robot to move toward the mark could be useful. Another module could calculates trajectory of moving objects to reach mobile marks.

References

[1] Michel JOURDIN, Métriques de Formes et Applications, 2002, école CPE Lyon, laboratoire d´élèctronique, traitement d´image et automatique, CPE Lyon.

[2] Robot Cup German team 2003 and 2004, http://www.germanteam.org/GT2003.pdf

[3] Raphaël Haberer-Proust, Semester project, 2005-2006, Remote control of AIBO camera from Webots, http://birg.epfl.ch/page59430.html

[4] Alessandro Rovetto, Francesco Scandelli thesis: Semi-Autonomous Navigation of a Legged Robot using Monocular Vision, 2005, http://www.sais.se/mthprize/2005/rovetto_scandelli2005.pdf

[5] Uwe Düffert Diploma Thesis, Quadruped Walking modeling and Optimization of Robot Movements, 2003 http://uwe-dueffert.de/publication/dueffert04_diploma.pdf

[6] Wikipedia: 3D projection, http://en.wikipedia.org/wiki/3D_projection

[7] David S. Touretzky and Ethan J. Tira-Thompson, Exploring Tekkotsu Programming on the Sony AIBO, Carnegie Mellon University, http://www.cs.cmu.edu/ dst/Tekkotsu/Tutorial/forwardkin.shtml

[8] Roger P. Woods, course on homogenous coordinates.

[9] Webots user guide: Using AIBO robots, cyberbotics.com