Humanoid Chess Player

Björn Franzon 811107-4871
quack@etek.chalmers.se
January 8, 2006 Gothenburg

Autonomous Agents FFR125 and
Humanoid Robots FFR155
Complex Adaptive Systems
Chalmers University of Technology

Abstract

Humanoid robots are knocking on the door to our world as this project of building a robot that look, act and play chess as a human. The robot is built in the same size as a human and is able to recognize the opponent's move, calculate a new move and point out the move to the opponent. The robot is able to beet a beginner in chess, sensitive to light and misses a piece with a maximum of three centimeters. Economy and time often had to govern what solution to take. This project shows that building robots for smaller tasks are possible with today's technology and I believe we soon can see a humanoid chess player in the stores.

Contents

1 Introduction

Humanoid robots have not yet become every man property but in just a few years there are no doubt that will change. The thought of applications where a robot would come in handy are starting grow in peoples mind and one important role that the robots will have is to be a good friend and play with us. The goal of this project is to:

> "Build a humanoid robot that can play chess against humans."

The reason of choosing to build a humanoid chess player is that the project involves many different fields that have to be synchronized in order to work. An important aspect is that the robot should be able to adapt as much as possible to our world instead of the opposite, which often is the case of today's robots. When I started this project I had problems knowing how far I would get and how much time different parts would take to work.

2 Design

2.1 Mechanics

To be able to build a robot that can play chess and look human the body is made in the same size as a human and only consists of the upper half of a human. To keep a low weight and stable construction the body is made of aluminum. The body is lending forward to make it look more natural when the camera is placed right above the chess board and also for the arm reach further without hitting the stomach. The bottom of the robot is made of wood with an extra weight to prevent the robot from falling forward. Figure 1 shows the design of the robot.

(a) Overview of robot (b) Starting position (c) Upper arm

Figure 1: Different view of the robot

2.2 Electronics

The only parts that are able to move are the servos which are controlled by a servo controller card. The servo controller card signal is pulse width modulated in order to tell the servos what angle and speed to use. The servos get there current from one source and the servo controller card from another to not have impact on each other. The servo controller card gets a message signal from the computer via the serial (or com) port but has to be transformed from 12 volt to 5 volt (max255) first. The electrical components are shown in Figure 2 and also listed in Table 3 in components Appendix.

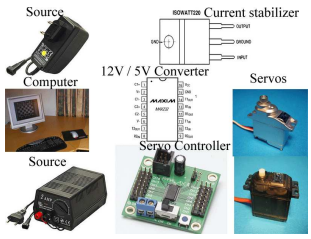


Figure 2: Electrical components

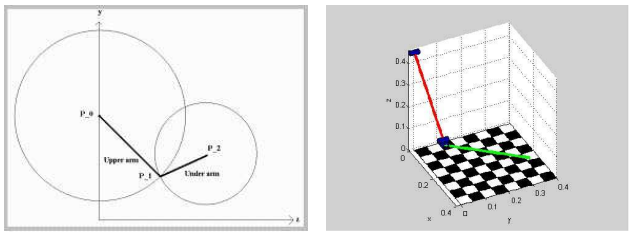
2.3 Chess board

The chess board is 30x30 cm and all 16 black pieces has an extra tape around them and all 16 of the white pieces has an extra black tape around them. The tape is there in order to make the image processing easier.

3 Control

3.1 Model

The arm consists of the two limbs upper and under arm which are able to move because of the servos. The upper arm is able to move in a half circle and the under arm in half a sphere. To be able to move the hand to a specific piece the angles for the servos needs to be calculated from a Cartesian coordinate system build from the chess board, this is also called the inverse kinematics. The inverse kinematics in this simple example with three degrees of freedom [1] was found in the geometry in Figure 1 and because of the limitations in the rotation only one solution for each position is possible. In the right part of Figure 3 P_0 is the shoulder and P_2 is the hand and because those are fix P_1 can only be at a point where the two circles meet.



(a) 2D-Model of robot (b) 3D-Model of robot

Figure 3: Two pictures of the mathematical model, to the left x-y-z and to the right y-z (different examples in the figure)

$$R_{Under} = \sqrt{L_{Under}^2 + P_{2x}^2} \quad (1)$$

$$Dist_{P_0P_2} = \sqrt{(P_{0y} - P_{2y})^2 + (P_{0z} - P_{2z})^2} \quad (2)$$

$$R_{Upper} = L_{Upper} \quad (3)$$

The final inverse kinematics can be seen in the equations below.

$$Servo1 = \frac{\pi}{2} + arctan(-\frac{P_{0z} - P_{2z}}{P_{0y} - P_{2y}}) + arccos(\frac{R_{Upper}^2 + Dist_{P_0P2}^2 - R_{Under}^2}{2 \cdot R_{Upper} \cdot Dist_{P_0P2}}) \quad (4)$$

$$Servo2 = arccos(\frac{R_{Upper}^2 + Dist_{P_0P2}^2 - R_{Under}^2}{2 \cdot R_{Upper} \cdot Dist_{P_0P2}}) + arccos(\frac{R_{Under}^2 + Dist_{P_0P2}^2 - R_{Upper}^2}{2 \cdot R_{Under} \cdot Dist_{P_0P2}}) \quad (5)$$

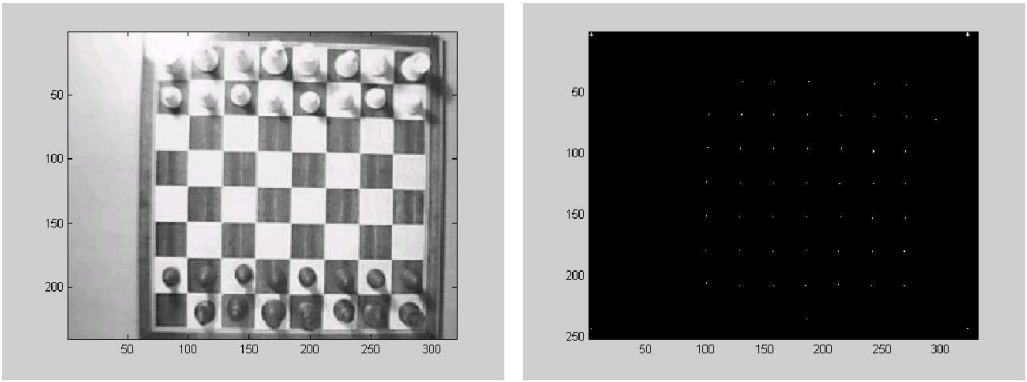
$$Servo3 = arctan(\frac{R_{2x}}{R_{Under}}) \quad (6)$$

3.2 Calibration

Before the inverse kinematics angles could be sent to the real servos they need to be transformed and tested against the real system. Some of the calibrated parameters are arm lengths, servo angles and position of the web camera. The web camera is also sensitive to direction and light intensity to make the image processing work.

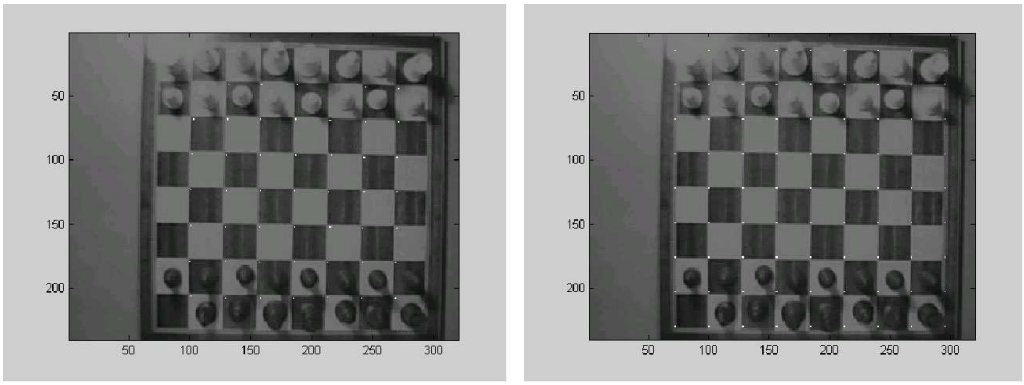
4 Image Processing

For the robot to be able to analyze the game a web camera is used as an eye [1]. The reason for choosing a web camera is because it is cheap, small and easy to implement in a running program. An image is captured every second with the format 320x240 and each pixel is stored as a gray scale integer value between 0 and 255. An example of how the image processing works can be seen in Figure 4 where 4(c) and 4(d) has a darker version of the input image in the background to show the result.



(a) Input image

(b) Chess square filter



(c) Symmetric filter

(d) Chess board location

Figure 4: Image Processing

The first step in the image processing is to use the chess square filter which was found by trail and error. The filter can be seen in Table 1 and it is used together with its transpose in the function "conv2" in Matlab (see Figure 4(b) for the result).

The next step is to set all pixels that are above a given threshold to one value (white) and the rest to another (black). White pixels that are close to each other are then reduced to only one white pixel. After this the mean distance to the closest white pixel are calculated and if one of the white pixels deviate to much from this mean value that one is taken away. The next step is to check if more than at least four white pixels, some might be missing, stay in the same width as well as four at the same height with a small deviation. The result of the symmetric filter is shown in Figure 4(c).

1	1	1	-1	-1	-1
1	1	1	-1	-1	-1
1	1	1	-1	-1	-1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1

Table 1: Chess square filter

The chess board location is than found by just adding another layer of squares outside the seven rows and the seven columns that are left (see Figure 4(d)). If the image processing found more or less than seven rows or columns a fault message is sent back to tell that the image processing failed to detect the chess board.

The final task for the image processing is to find the pieces in the calculated squares. In order to make it simple the image processing only detects if a piece is placed inside a square and if that piece is black or white. Another thing that was made to make the image processing simpler was that all pieces were given an extra bit of tape, black tape to white pieces and vise versa. The reason with the tape is that an edge detecting filter and a threshold can detect if a piece is placed in a square or not even if a black piece is placed in a black square or white in a white square. The edge detection filter used in the piece detection part is seen in Table 2.

1	2	1
0	0	0
-1	-2	-1

Table 2: Edge detection filter

5 Chess Computer

The chess computer is divided into one class that handles the rules, one that display the game in a graphical interface and one that play as an opponent. A snapshot from the game is shown in Figure 4.

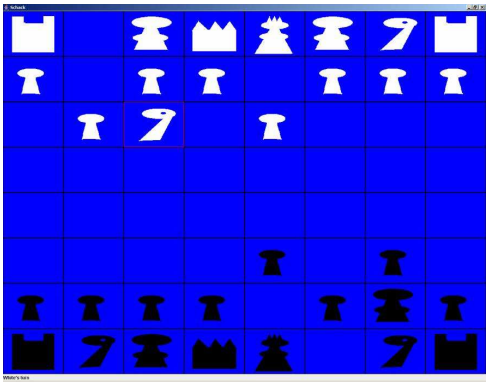


Figure 5: Snapshot from the graphical interface for the chess computer

5.1 Rules

The rules of chess are simple and one state could be described by just knowing whose turn it is, if the kings still can make castling and of course where the different pieces are placed on the board. Different pieces can move in different direction and with different length and by being able to take your opponents king without the opponent being able to prevent you from doing so is the only way to win or vise versa to lose. If the person that has to move can not move and is not in check or if no one can win with the pieces that are left the game ends as a draw.

5.2 Graphics

The graphics interface display a given game state and tell if a player is in check and if the game is over. The graphics can also be controlled from the web camera or by a mouse click to play a game between two humans or against the chess computer with or without the robot connected.

5.3 Opponent

The chess computer uses the rules class to find all valid moves. Each move are then given points depending on how good they are and what the opponent could do if this move is made. The algorithm is a so called minimax method [1] which minimizes the opponent maximum move. The chess computer also has a few stored openings which it randomly selects from in the beginning of every game.

6 The Brain

The robot is controlled by one thread so it always finishes one task before starting another. The only thing that the robot keep in mind at all time is the actual game state so it has to start the image processing and chess computer all over every time. The limitation in speed lies in keeping a stable movement of the arm and in the image transfer from the web camera to the computer. An information flow diagram is shown in Figure 6. Everything is implemented using Matlab 7.0 except the chess computer which is implemented in Java.

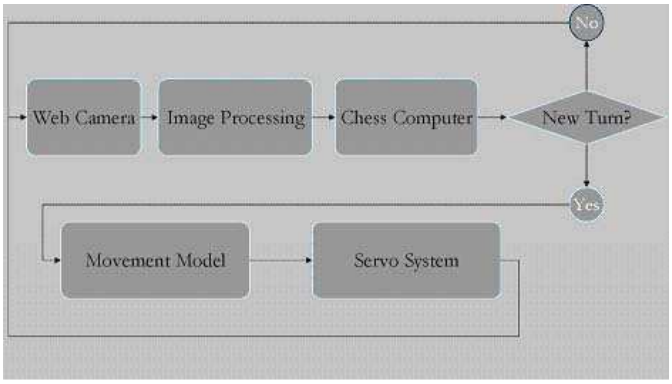


Figure 6: Information flow

7 Results

It is difficult to know how much time it takes to build a real robot rather than just simulating one in a computer. For a smaller project like this one the economy often had to govern what solution to take. The servo controller card is sensitive to large current changes as when the arm is working hard or when the power is turned on. The mathematical model was simplified in order to get easier calculations and servo expressions. After calibration of the chess board location the mathematical model missed the correct position with a maximum of 3cm. The image processing is sensitive to chosen web camera resolution, light intensity as well as shadows falling on the chess board. The chess computer is able to beet a beginner in chess.

8 Discussion

This project clearly shows that building robots for smaller tasks are possible with today's technology. The reason why robots have not yet reached the market could be that we here in Europe are a bit frightened of the term robot. I think robots will become a market of the same size as the car industry and that it is just a question of time before we can find singing, board game playing, vacuum cleaning and home help service robots in the stores.

9 Conclusion

Because there are only 64 squares in a chess game it is possible to keep the chess board at a specific distance from the robot and by that hard code each position in order to get a better precision.The chess square filter works well if the entire chess board is shown in about 250x250 pixels, so if the resolution changes the implemented image processing probably fail to find the table. There are also many thresholds that are set by hand which makes the detection sensitive to the light intensity. The image processing could be improved to detect different pieces by a more advanced algorithm or by using different markings to each type of piece but that would be a lot more difficult to achieve. The edge detection method is sensitive to shadows which makes it hard to find a working illumination.

10 Future Work

For the robot to function as a humanoid chess player it needs to be able to lift the pieces and hopefully doing that in a human like way. The preferred solution is to lift the pieces by grasping fingers. A not so human like but easier way would be to use electro magnets, but this would mean that the pieces must be modified which should be avoided if possible. One thing that has to be done before the robot could function properly is to reach all pieces. This could be done by placing an extra servo in the shoulder of the robot. A commercial "humanoid chess player" could have an inbuilt computer, loudspeakers for talking ability and a nice design with soft materials and a human like head.

11 Thanks

I would specially like to thank my good friend Almir Heralic who made the whole project possible for me to carry out. Also I would like to thank my teachers Krister Wolff and Anders Eriksson for there help.

A Appendix: Code

```
% Some useful Matlab code for serial port, webcamera, java and servo control

main.m
% Starting the serial port
if(1) % 0 if serialport aready active
  init_serialport;
  if( ~isvalid(serial_port) )
    error('Serial Port not connected');
  end
else
  serial_port = instrfind;
end

% Connecting a web camera (Matlab 7.0 is required)
vid = videoinput('winvideo', 1);
preview(vid);
% get image from webcam
I1 = getsnapshot(vid);
% from rgb to gray
I=im2double(rgb2gray(I1))*255;

% Connecting Matlab to JAVA
javaclasspath({'..\chess_comp'});
CR=javaObject('ChessRules');
CR.printGame;

% Moving the robot arm to a specific point
P_1 = [20, 40, 10]; % point with vaues for x-y-z
S=invers(P_1); % Calculate the invers kinematict for the servo angles
ramp=17; % "Rampage" Set speed of movement
S_CARD_POS=[1,3,5]; % Servo number on the servo controller card
moveRobot(serial_port, S_CARD_POS(1), ramp, S(1)); % Move first servo
moveRobot(serial_port, S_CARD_POS(2), ramp, S(2)); % Move second servo
moveRobot(serial_port, S_CARD_POS(3), ramp, S(3)); % Move third servo

get_bytes.m
% Transform a value to byte form
function [lowbyte_decimal, highbyte_decimal] = get_bytes(value)
binvector = dec2binvec(value,16);
lowbyte = binvector(1:8);
highbyte = binvector(9:16);
lowbyte_decimal = binvec2dec(lowbyte);
highbyte_decimal = binvec2dec(highbyte);

moveRobot.m
% Write to the serial port to send information to the servo controller card
function moveRobot(serial_port, servo_nr, ramp, position)
[lowbyte, highbyte]=get_bytes(position);
fwrite(serial_port, char(23,83,67,servo_nr,ramp,lowbyte,highbyte,13)');
```

B Appendix: Components

Product	Name	Bought From	Article Number
Voltage Source	Power Supply 3-12 V	BILTEMA	38-114
Voltage Source	Voltage Adapter	BILTEMA	38-110
Servo	BMS-380 MAX	MFT	BMS-380 MAX
Servo	BMS-620MG	MFT	BMS-620MG
Servo Controller Card	Parallax	ELFA	73-196-07
12V / 5V Converter	MAX232	ELFA	73-023-26
Voltage Regulator	7805	ELFA	73-262-75
Digital Web Camera	Creative	SIBA	N10225
Aluminium Material		Metallvaruhuset AB	

Table 3: Components

References

[1] Stuart Russell and Peter Norvig, *Artificial Intelligence A Modern Approach*. Prentice Hall, New Jersey, Second Edition, 2003, page 165 863-874 904.