Implementation of an autonomous chess playing industrial robot

Jens Golz[1], Rolf Biesenbach[2]
Faculty of Electrical Engineering and Computer Science
Hochschule Bochum - Bochum University of Applied Sciences
Bochum, Germany
[1]jens.golz@hs-bochum.de, [2]rolf.biesenbach@hs-bochum.de

Abstract* – This paper describes how a standard 6-DOF industrial robot is equipped with additional hard and software to enable the machine to play chess with a human chess player. For automatic interaction with human player, these moves were recorded by a webcam and automatically analyzed. No manual (keyboard-) input is necessary. One goal of the work is that the implementation of the chess playing robot should be done by using standard software to minimize the use of self-written code. Due to that target the controller of the robot interacts with an external computer, hosting MATLAB, the KUKA-KRL-Toolbox for Matlab and the Open-Source chess engine STOCKFISH. All software components which are not supported by the origin robot system, are integrated by using MATLAB. Introduction

Keywords: autonoumous chess playing robot, *KUKA robot controller, KRC2 ed05, KUKA KR16, mechatronics*

I. INTRODUCTION

Due to reasons of operator safety and protection a lot of manufactures of industrial robots are restricted in their range of supported hard- and software. Using a software like MATLAB, makes it comfortable to integrate not supported new hard- and software components in the robot system. This combination creates an amount of new opportunities expanding the robot system and its functions. These new possibilities are shown in this project.

By the reasons described above, the unrestricted principle of "Plug & Play" is not applicable to the used robot controller. A bridge like the "KUKA-KRL-TBX" is needed to implement the project. "KUKA-KRL-TBX" is a MATLAB toolbox in order to connect and control a KUKA industrial robot with an external remote PC. This toolbox was developed by a research group from Wismar University of Applied Sciences [1].

That leads to the following partition of the implementation. The original KUKA robot controller, KRC2 ed05 is used for the robot motion control and an external computer serves as a host for the chess program, the pattern recognition and additional tools. Here also the webcam is integrated and by the use of the MATLAB toolbox "Image Processing" the operator can handle complex applications

This contribution demonstrates how the functions are implemented. The Open-Source chess engine "Stockfish" [2] is used to determine the robot's turn. Stockfish subjects to the GNU General Public License (GPL) [3] and therefore it is freely available and distributable, it also works with the Universal Chess Interface (UCI) protocol [4], which was designed for chess engines to interact with Graphical User Interfaces (GUI's). The Stockfish chess engine is built on an algorithm which uses a database of records from matches of professional chess players to compare the done turns and return the best next move you can do. To name the chess moves the algebraic chess notation (ACN) [5] is followed. ACN is the most commonly used method for recording the moves of a game of chess. The eight columns get named with a – h and the eight rows get called with 1 – 8 as seen in Fig. 1.

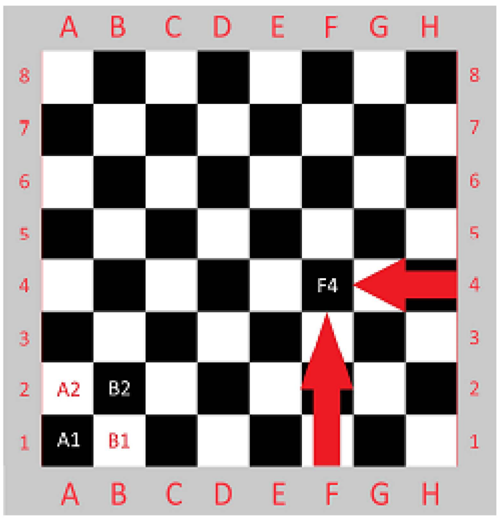

Fig.1 Algebraic chess notation [6]

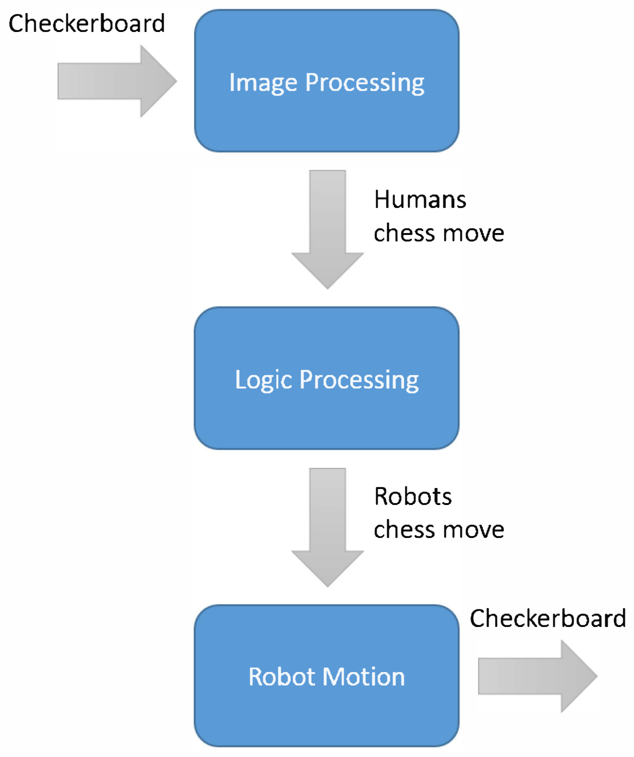

Fig. 2 Partitions Illustrated

II. Analyzing / Image processing

To capture the human's chess move the webcam records two snapshots of the checkerboard. One picture before and one after the human's move is done. Then the MATLAB Image Processing Toolbox [7] function "imfuse" with the parameters (image1,image2) builds the composite of both pictures. The equal sections of these pictures will be shown in gray and the different parts will be, depending on whether the brightness changed from light to dark or vice versa, shown in green or red. So the moved chessmen will be shown colored on an otherwise gray background. By using the MATLAB Computer Vision System Toolbox [8] object "vision.BlobAnalysis" it is possible to detect these colored blobs.

Before these blobs could get detected the fused image has to be divided into its color planes, Red (R), Green (G) and Blue (B) and the images of these color planes have to be converted into a binary black-white image.

By coloring every pixel below a specific brightness threshold into black and those above into white, it will be possible to use the "BlobAnalysis".

In Fig.3 the function of the Image Processing Partition is shown with original images. This example shows the detection of the red color planes blob, to get the information about the complete chess move, the green color plane have to be inspected additionally.

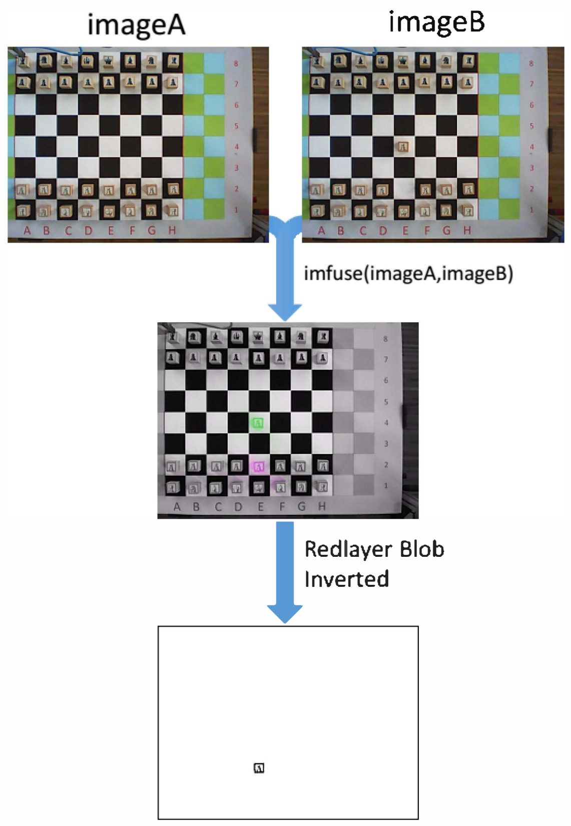

Fig.3 Image Processing Function

By using a "step()" with the "vision.BlobAnalysis" object and the prepared fused image as parameters the area centroid and bounding box of the blobs is calculated. To get the blob's centroid position into a for the chess engine understandable format, a pixel interpreter is used to convert it in a chess notation conform representation. The operation of the pixel interpreter is illustrated in Fig. 4. The positions of the two blobs will be merged and given to the next part as "hTurn". If a piece moves from a black to a white field both blobs will be shown in the same color. In this case the result of "step()", the returned array of blob information, will have an additionally row, with the information of the second blob.

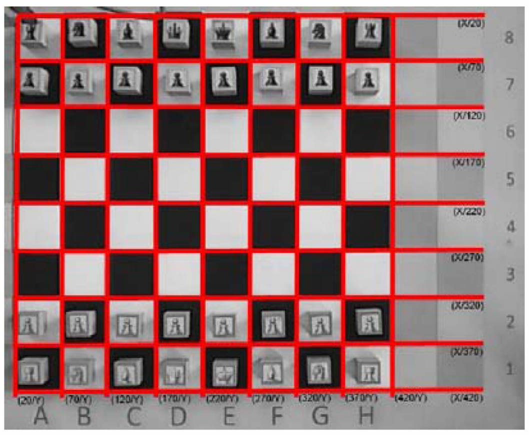

Fig. 4 Operation of the pixel interpreter on frame grab

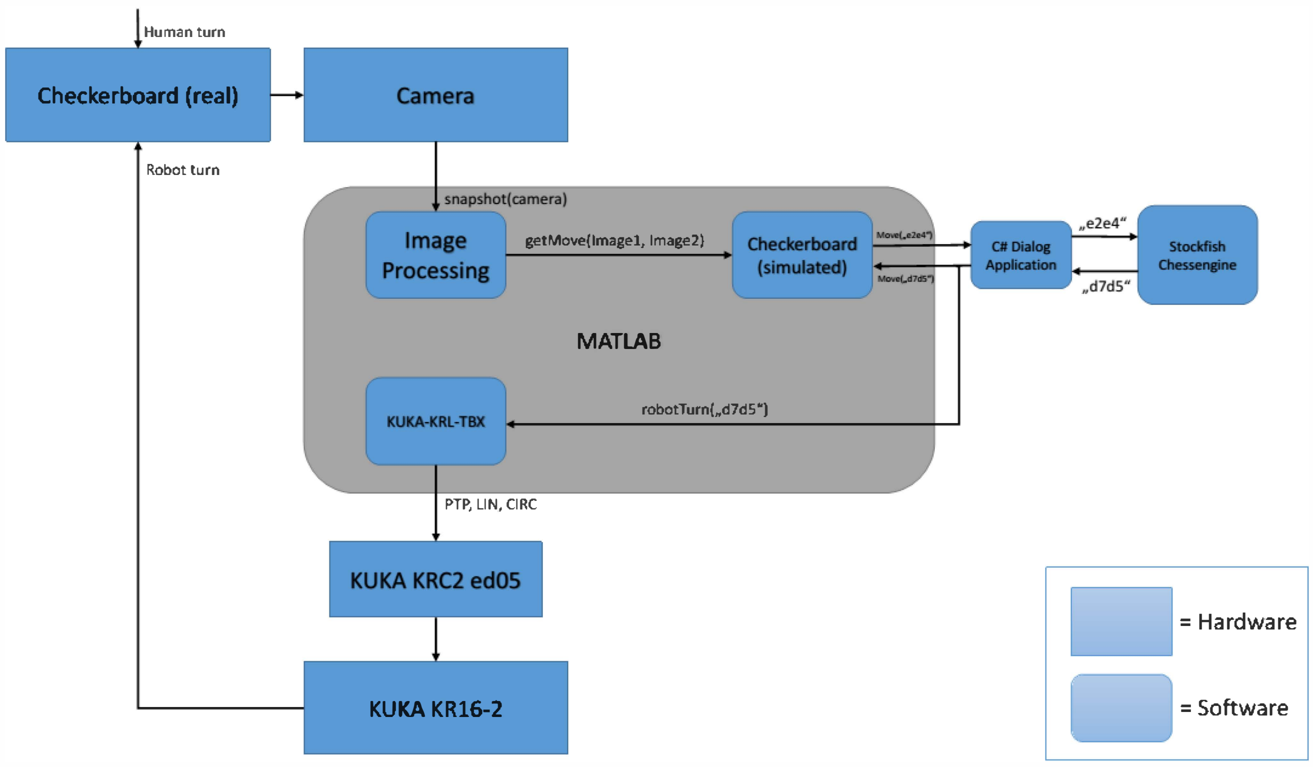

Fig. 5 Program flow diagram

III. SIMULATION / LOGIC PROCESSING

In order to make the robot able to respond on the human's chess move, the public chess engine STOCKFISH has been integrated.

The automated algorithm operates as follows:

1. First the system checks the human's move.
2. When the human chess opponent has moved a correct pawn (Block, which stands for a chess piece) and especially moved it correctly according to the chess rules, the whole (digital) checkerboard gets refreshed.
3. Every chess piece got a logical expression for a correct move. Additionally for those pieces which move along a line, like the bishop for example, it has to be checked that there is no other piece on this line.
4. The simulated checkerboard is a two dimensional integer array with the size of 8x8 and it is needed to use the biuniqueness of the algebraic chess notation. All done moves are stored in the variable "turnHistory". This turnHistory is a string with all the collected turns, which will be given to the Stockfish chess engine.
5. As a result, the chess engine calculates the best next move and saves it in the variable called "rTurn". Again the simulated checkerboard gets updated with the new positions. And the recommend move will be also given to the next partition robot motion.

IV. ROBOT MOTION

To handle the robot's turn, the function "robotTurn(rTurn)" is used to calculate the desired robot arm positions and handles the pick and places procedures of each piece. It calculates the desired moves from chess notation into Cartesian coordinates. The robot arm got a fixed starting position on the field (A8), from where it acts with relative movements. The actual robot arm movement is realized by the "KUKA-KRL-TBX" [1]. The toolbox uses all common sorts of movements, including Point-to-Point (PTP), Linear (LIN) and Circular (CIRC). To transfer the signals, a RS-232 serial connection is used. This connection is pretty robust but slow, so that real time applications can't be realized.

V. Conclusion and future work

By using a communication toolbox to connect a KUKA robot controller with MATLAB a program has been implemented which allows the robot arm to play chess against a human being. A webcam and a chess engine got integrated, into the MATLAB program, for making the robot recognize the human player's turn and calculate the next best move to win the game. An important part of the program is the algorithm that checks if the turn, the human player made, conforms to the chess rules. This has been realized with multiple functions which includes the mathematical expressions of each legal move. The connection between the chess engine and MATLAB was realized by a C# application. This application reduces the communication to MATLAB on the transfer of the coordinate movements and communicates to the chess engine according to the UCI protocol.

According to the main idea, the project is intended to show the possibilities of robot intelligence. There are some suggestions to improve the project in the future, as the following:

- Actual the operator has to keep the KUKA Control Panel (KCP) in his hand, while he is playing against the robot, to press and hold the enable key. To avoid this, a future project could be to renewal the Safety Concept of the robot cage. A smart combination of a kind of chess clock and a laser fence could replace the metal door to allow run the program in automation mode. With an extensive risk analysis the legitimacy should be proved.
- The Checkerboard and the chess pieces could be adapted to the tournament standard. For this the gripper and the pixel interpreter have to be adjusted.
- The image processing could be also improved by robust board recognition.
- In 2015 a new toolbox for the communication between the actual KUKA KR C4 controller and MATLAB is developed at the Faculty of Electrical Engineering and Computer Science of Hochschule Bochum [11][12]. The next stage of development is, to adapt the project presented to the newly developed MATLAB-KR C4 interface and the actual KR C4 controller.

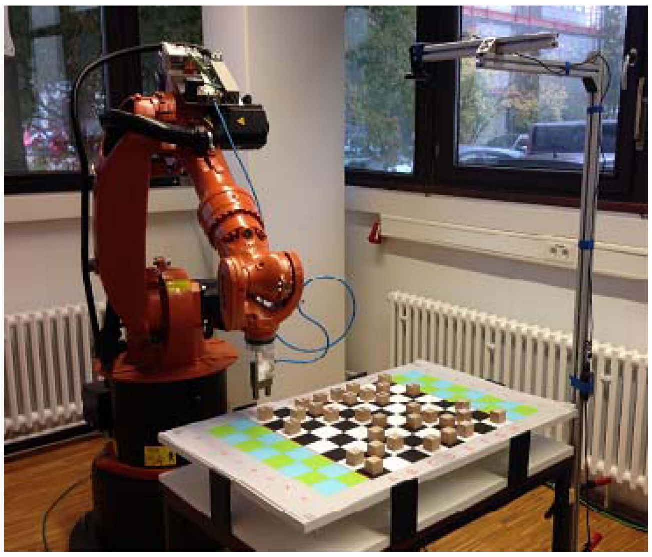
Fig.6 Chess playing KUKA KR 16-2 robot

References

[1] G.Maletzki, M.Christern, A.Schmidt, T.Pawletta, P.Dünow, "KUKA-KRL-Toolbox for Matlab® and Scilab (MatlabKukaKRL-Toolbox)". Wismar University of Applied Sciences: 2015, [Online]:http://www.mb.hs-wismar.de/cea/Kuka_KRL_Tbx/ Kuka_KRL_Tbx.html3

[2] Tord Romstad, Marco Costalbam, Joona Kiiski. "Stockfish Open-Source-Chess Engine": 2015, [Online]: https://stockfishchess.org/

[3] License, GNU General Publc. "GNU General Public License, Version 3": 2007, [Online]: http://www.gnu.org/licenses/gpl-3.0.de.html

[4] Stefan Meyer-Kahlen, Rudolf Huber. UCI Schach Engine Protocol. Shredder Chess. 2015. [Online:] http://www.shredderchess.de/download.html

[5] Chess notation, Wikipedia: 2015 [Online]: https://en.wikipedia.org/wiki/Chess_notation

[6] Figure, "Algebraic Chess notation" Wikipedia: 2015 [Online]:http://en.wikipedia.org/wiki/Algebraic_notation_%28chess%29#/media/File:SCD_algebraic_notation.svg

[7] Toolbox, Mathworks Inc.. "Matlab® Image Processing Toolbox description and examples" Mathworks Inc.: 2015 [Online]: http://www.mathworks.com/products/ image/index.html?refresh=true&s_tid=gn_loc_drop

[8] Toolbox, Mathworks Inc.. "Matlab®Computer Vision System Toolbox, vision.Blobanalysis description and examples". Mathworks Inc.: 2015, [Online]:http://www.mathworks.com/help/vision/ref/vision.blobanalysis-class.html

[9] Hahn, B., & Valentine, D. (2007). Essential MATLAB for engineers and scientists. Newnes.

[10] KUKA Robot Group (2007), "CREAD/CWRITE, Programming CREAD/CWRITE and related statements"

[11] Implementation MATLAB Toolbox for motion control of KUKA KR6-R900-SIXX robotic manipulator, H. Elshatarat, Hochschule Bochum, 2015

[12] MATLAB Toolbox Implementation and Interface for motion control of KUKA KR6-R900-SIXX robotic manipulator, H. L. Elshatarat, R. Biesenbach, Hochschule Bochum, Germany, M. Bani Younus, T. A. Tutunji, Philadelphia University of Jordan, Jordan, Proceedings REM2015, 2015