A Simple Autonomous Robotic Manipulator for playing Chess against any opponent in Real Time

Nandan Banerjee, Debal Saha, Atikant Singh, Gautam Sanyal

Nandan Banerjee (email: banerjen@hotmail.com) and Atikant Singh are undergraduate students with the Dept. of Computer Science and Engineering, Debal Saha (email: sahadebal@gmail.com) is an undergraduate student with the Dept. of Mechanical Engineering and Dr.Gautam Sanyal (email: nitgsanyal@gmail.com) is a professor with the Dept. of Computer Science and Engineering at the National Institute of Technology, Durgapur, INDIA. The project was funded by the Dept. of Computer Science and Engineering, National Institute of Technology, Durgapur, INDIA and manufacturing of the manipulator was done at Senco Engineering Works, Durgapur, INDIA.

***Abstract*— This paper presents a simple 3-DOF (degree of freedom) robotic serial manipulator which is capable of playing chess in real time against any opponent. This autonomous chess playing robot consists of a HD Logitech Webcam, a low cost custom made serial manipulator, algorithms for the efficient detection of the chess pieces on the chessboard and a robust control mechanism for the accurate movement of the manipulator**

***Keywords*— Human Robot Interaction, Chess playing robot**

I. Introduction

This project introduces a 3 DOF robotic manipulator system that is designed to autonomously play the game of chess in real time against human or robotic opponents. The Chess Robot includes a Logitech HD webcam, a low cost custom made manipulator, algorithms for the efficient detection of the chess pieces on the chessboard and a robust control mechanism for the movement and control of the manipulator. Compared to prior work on robotic systems playing chess with specifically instrumented chess boards and/or pieces, our robot is not confined to just playing chess but it can also be used as a standard manipulator and can be controlled effectively using its own set of control commands.

II. Chess Piece Detection

The chess pieces on the chessboard were detected using standard image processing techniques.

A. Prerequisites

For the image processing part in which the main objective was to find the current orientation of the board given the previous orientation, the following were used:

- openCV library (used along with qt, an open source C++ framework)
- Logitech C270 webcam
- Intel Core2Duo processor running at 1200 MHz

B. Methodology

A webcam was positioned directly over the chessboard and the resolution was set at 640 * 480. The openCV image processing library was used to capture the screenshots and apply the image processing procedures. Following factors were identified as having a major effect on the performance of the image processing techniques:

- Texture/pattern of the chessboard
- Colour of the squares/pieces
- Lighting conditions

Therefore, the colours of the squares on the chessboard were switched to red and faint lemon yellow to offer a better contrast than the earlier black-and-white chessboard with black-and-white chess pieces on it. The most suitable light source was found to be a diffused light source on to the surface of the board. When a pointed light source from the side was found to have been used, shadows and glares thus created were sometimes misinterpreted as pieces.

C. Image processing algorithm

1) Set the Region of Interest: A screenshot is taken using the webcam and the region of interest is set manually. The region of interest is the part of the image where all the image processing procedures are to be applied.

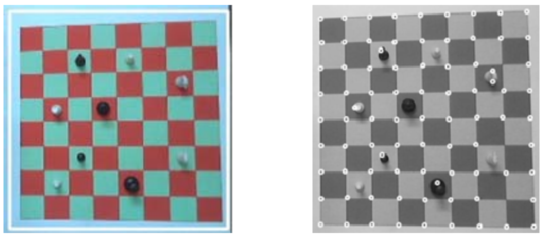


Fig 1. (a) Region of Interest around a screenshot of the chessboard. (b) All probable corners detected using the Shi-Tomasi corner detection algorithm.

2) Detect the corners and intersection points: Using the Shi-Tomasi corner detection algorithm, which is just a slight modification of the Harris corner detection algorithm, the corners of the chess squares are found out.

3) Find the top-leftmost, top-rightmost, bottom-leftmost and the bottom-rightmost corners: The four endmost corners of the chessboard are then determined using normal pixel position calculations.

4) Compare the probable and the actual corners: The probable corners are found out by dividing the difference of the corner points by eight, thereby finding out the corner points of all the chess squares. Then the probable corners are compared with the actual corners and if the difference between their values suggests that they are neighbouring points, then the probable corner is replaced by the actual corner. In this way, the corners for all the 81 chess squares are thus found out. This method is used to remove the corners that are detected on the chess pieces. Also, because of bad ambient lighting conditions, sometimes a few of the corners are not detected. In that case, the probable corner acts as the corner of the chess square. Therefore, to avoid these two errors, this comparison and assertion procedure is performed.

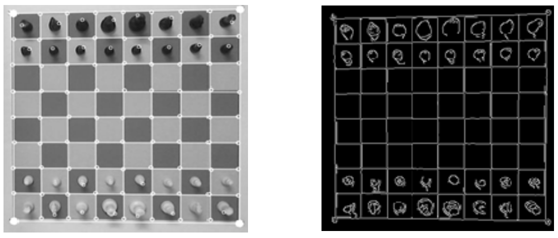


Fig 2. (a) Showing all the chess squares with their corners detected. (b) After applying the Canny edge detection algorithm to detect all the chess pieces on the chessboard.

5) Check for the square occupancy using the Canny edge detection algorithm: The Canny edge detection algorithm is then used to detect the edges of the chess pieces on the chessboard. From the already found values for the four coordinates of all the 64 chess squares, all the chess squares are chosen. The dilation operation is then applied on the edge detected binary image on each of the chess squares, and the number of white pixels is then compared with a designated threshold value. If it exceeds the threshold value, then the chess square is occupied otherwise empty.

6) Determination of the colour of the chess piece: The image is thresholded. The threshold is chosen according to the ambient lighting conditions and the texture of the chessboard. After thresholding, the black pieces stay black whereas everything else becomes white. Then, the erosion operation is applied on the thresholded image on each of the chess squares, and the number of black pixels is then compared with a designated threshold value. If the value exceeds this threshold value, then the chess piece is black otherwise white if the square is not otherwise empty.

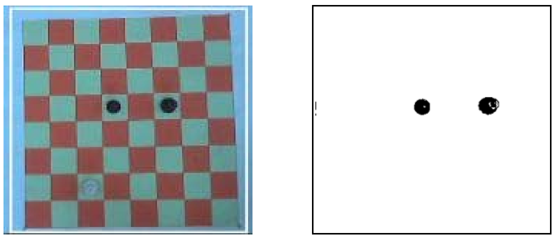


Fig 3. (a) Showing a chessboard orientation with two black pieces and one white piece. (b) After thresholding, only the black pieces are deemed as objects and everything else is ignored.

D. Chess Move Detection

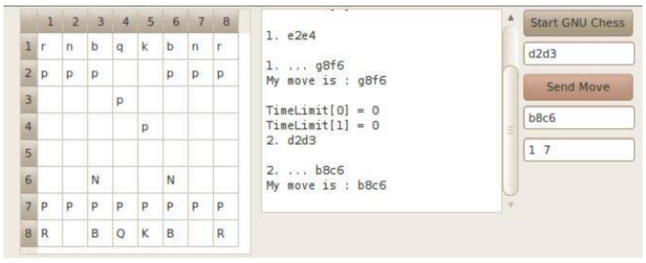
After the orientation of the chessboard has been successfully detected, the individual identities of every chess square is known, i.e. if the chess square is empty or not, and if not empty, then the colour of the chess piece present in it. An eight by eight matrix is then constructed consisting of three values [0, 1 and 2]. '0' signifies that the square is empty. '1' signifies that the square has a white piece in it and a '2' signifies that the square has a black piece in it. The image processing algorithm described above is applied on the entire chessboard and all the 64 values of the squares are hence determined and stored in the matrix. Then the matrix is compared to a previously recorded matrix of a previous image and the changes are noted. From these changes, it can be ascertained as to what move has been given by the opponent. After this move is realized, it is then fed as the input to the GNU Chess engine described in the next segment. The vision system could be thwarted simply by swapping chessmen so it is being assumed that there won't be any foul play on the user's part.

III. GNU Chess Interface

GNU Chess is a computer program which plays a full game of chess against a human or other computer program. The goal of GNU Chess is to serve as a basis for research. It has been used in numerous research contexts. It is free software, licensed under the terms of the GNU General Public License and is maintained by collaborating developers. It's one of the oldest computer chess programs for UNIX based computers and one of the earliest available with full source code.

On basic, current computer chess architecture, GNU Chess plays at senior master/weak international master strength (2500+_ELO on simple hardware – Intel Core2Duo), without parallel processing, according to the IQ6 test suite.

In this project, the GNU Chess version 6.0.1 has been used to determine the moves by the Chess Robot. The GNU Chess engine has been interfaced in "qt"` using the QProcess class. GNU Chess is started in the Xboard mode using the -x argument. The moves are sent to and received from the GNU Chess engine using standard coordinate algebraic notation through the standard input and output streams.



	1	2	3	4	5	6	7	8
1	r	n	b	q	k	b	n	r
2	p	p	p			p	p	p
3				p				
4					p			
5								
6			N			N		
7	P	P	P	P	P	P	P	P
8	R		B	Q	K	B		R

```
1. e2e4

1. ... g8f6
My move is : g8f6

TimeLimit[0] = 0
TimeLimit[1] = 0
2. d2d3

2. ... b8c6
My move is : b8c6
```

Fig 4. GNU Chess Interface using QProcess in qt.

As mentioned in the previous section, the move ascertained after applying the image processing techniques is fed into the input stream of the GNU Chess engine which is interfaced using the QProcess class as already mentioned. The GNU Chess engine then figures out the appropriate response and it is then relayed to the robot control module which then automates the robot.

IV. Mechanical Aspects of the Serial Manipulator

The various mechanical aspects of the serial manipulator can be categorized as follows:

- Mechanism Development and Actuation
- Stress and Force Analysis and CAD modelling

- Material Selection
- Manufacturing of the Serial Manipulator

A. *Mechanism Development and Actuation*

The manipulator is intended to perform a pick and place task. A simple model for a serial manipulator for this purpose is a 2PR (two prismatic and one rotary joint) serial manipulator. The base link carries the first link and there exists a rotary joint between them. Between the first and second and between the second and third link there are prismatic joints. While the prismatic joints help in easily locating the end-effector of the serial manipulator in the desired point in the work space, the rotary joint increase the total work space and provides more manoeuvrability.

The rotary joint between the base and first link is established by fixing a high torque motor to the base and mounting the first link on the motor shaft. The actuation of the prismatic joints can be achieved by using linear motors. But due to its high cost, an efficient and comparatively cheaper screw joint mechanism actuated by a dc motor is used to achieve the linear motion.

B. *Stress and Force Analysis and CAD Modelling*

The arrangement of the last link makes it act like a cantilever beam. The forces acting on the last link are same as that on a cantilever beam. The length of the last link is decided by the farthest distance needed by the end effector to traverse (the distance to the farthest square on the board). The links are linearly actuated by the screw joint motion for which they need to be cylindrical and shape and should carry screw threads. The pitch of the screw thread is decided by the velocity of the actuating motor and the required linear velocity. This cantilever beam type link is mounted on another link which too carries another prismatic joint which is actuated by a dc motor using the screw mechanism. The diameter of this cylindrical link is decide by calculated the cumulative effect of the bending moment and torsion of the cantilever beam on it and also the torsion due to its own inertia. This cylindrical link which supports the cantilever beam type link is mounted on a motor's shaft fixed to the base.

The stress and force analysis helps to decide the actuation torque required for the motors to drive these joints. The cantilever beam type link required a motor with comparative larger shaft length. The base motor needed a high torque motor as it had to bear the weight of the links mounted on it. the problem of increased torque due to the inertia/weight of the links on the base is resolved by mounting the links via bearing on the base and then actuating the link from the top end instead of mounting the whole link arrangement on the shaft of the motor and revolving it on the shaft.

The serial manipulator consists of various joints and links. The actuating joints need to be placed properly for dynamic balancing. The shape of thread, its position on the cylindrical links, the position of the motors, the mounting clamps, the alignment of the motor shaft etc. are some of the various points that are to be kept in mind while designing the manipulator. Most of the time the stress and force analysis is just not enough for obtaining the correct design which is free from all types of faults. Many decisions regarding alignments and positioning of various parts need to be decided in a heuristic manner. Computer aided designing packages help one to develop such designs. CATIA is one such package and it has been used to develop the design of this serial manipulator. The CAD designing approach helps us to perceive the implemented mechanisms in a better way. The probable areas of fault can be figured out while studying a CAD model and it aids in making fine changes in the intricate details of the model.

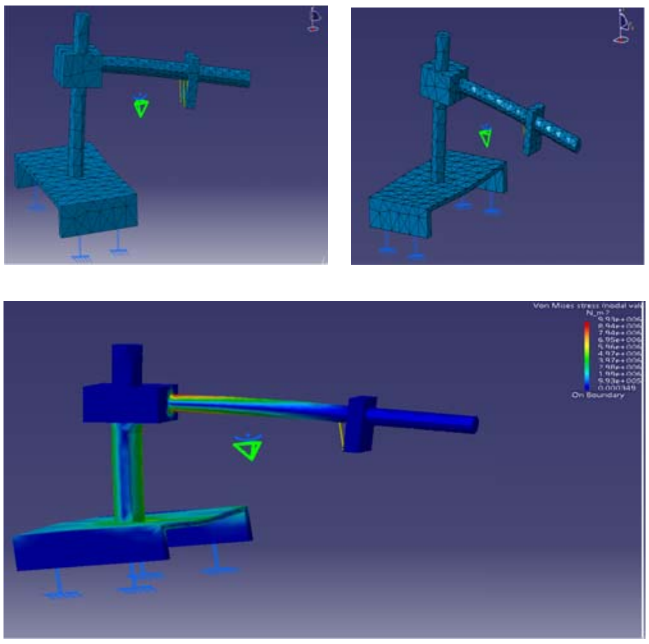


Fig 5. In the above figures, a simple representative model for the manipulator is used to analyse the stress distribution using the "Generative Structural Analysis" module of the CATIA software. In the clockwise direction, first the force applied and the mesh created for analysis is shown followed by the depiction of stress distribution with different colours representing various ranges of stress.

C. *Material Selection*

Once the mechanism is decided and the resulting force and torques are determined, the materials are selected for individual links of the manipulator. The base has the support the whole weight the other links and the actuating joints. Cast iron suits the requirements for such a base. It is capable of absorbing shocks. The base made of cast iron reduces the vibration of the cantilever beam type link and thus helps in precise positioning of the end effector. The first link on the manipulator has to support the cantilever type link and thus have to act like a strong column. The hollow cylindrical shape is suitable for this purpose. The thickness is limited by the machining stresses induced while making the threads on the column. Mild steel is suitable for making the column. The cantilever type link too is a hollow cylinder with low weight. Aluminium is a suitable material for manufacturing it to account for the low weight constraint.

D. *Manufacturing of the Serial Manipulator*

While manufacturing the manipulator certain issues were kept in mind. The selection of proper material is just not enough to dampen the inherent vibration produced in the links. The proper positioning of the actuating mechanism, the alignment of the links and mounting of the motors also play a crucial role in dampening this vibration and thus helps to attain dynamic stability and consequently better performance. The link which acts as the supporting beam for the cantilever beam type end link is mounted on a bearing on the base. The bearing is welded to the base and a protrusion from the supporting link is press fit in to the bearing. The cantilever type link has a cuboidal end with a cylindrical hole carrying inner groves matching the threads on the supporting beam. The cuboidal end has a protrusion which runs in a straight grove on a bar mounted on the base. This helps to attain a prismatic joint's to and fro motion with the help of the screw joint motion. Similarly the cantilever type link is fitted with a cuboidal block carrying a cylindrical hole with grooved inner threads that matches the threads on the cantilever type link and also possesses a protruded edge that runs on a grooved overhead bar. This cuboidal block that moves on the cantilever beam type link carries the end effector.

V. Gripper, Structure and Movement of the Manipulator

The manipulator has three degrees of freedom. The manipulator consists of 4 links and 3 joints. There are base linked can be termed as link 0 and the consecutive links can be numbered/named successively as link 1, 2 and 3. Joint 1 exists between link 0 and 1 and is a revolute joint. There exists two screw joints, one between link 1 and 2 and the other between link 2 and 3. The screw joints are good replacement for prismatic joints. The linear motors used for making a prismatic joint between to links is far more costly than the combined cost of machining threads (outer and inner) on links and that of a servo motor on which the links will be mounted after machining to form the screw joint.

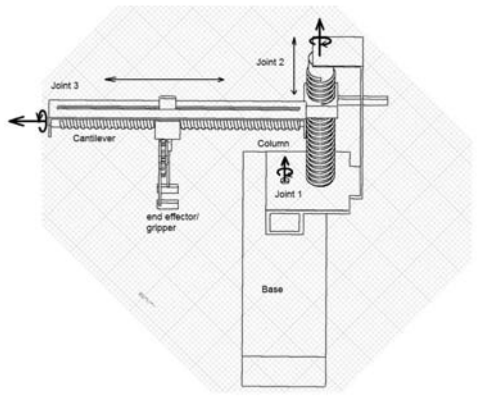


Fig 6. A sketch of the serial manipulator showing all the joints and links.

The linear motion achieved by the screw joint is also shown in the above diagram. The table/stand shows the base. Vertical column with the grooved mounting consists link one. The horizontal cantilever beam can be considered the link two. The end effector is the gripper which is in form of two spur gears with the gripping plates mounted on them. The revolute joint between the base and the first link is also shown. Thus, the joints and links are named as follows –

- Link 0 – Base
- Link 1 – Column
- Link 2 – Cantilever
- Link 3 – End effector/gripper
- Joint 1 – Revolute Joint
- Joint 2 – Screw (vertical) Joint
- Joint 3 – Screw (horizontal) Joint

Joint 2 is responsible for vertical lift necessary for picking up the chess piece and go for the necessary movement once of has been grabbed properly by the gripper. Once the image processing of the grabbed images of the board and chess pieces on it are done, a move for the serial manipulator is decide and the commands for the necessary movements of the joints for execution of the decision are transferred to the microcontroller. It thus drives the servo motors in compliance with the commands generated. The initial task for the manipulator is always to reach the desired coordinates where it has to pick up the desired chess piece. The picking up of the chess piece is the next task. Then it needs to take the chess piece to the destined position and then place it back on the board at the new position.

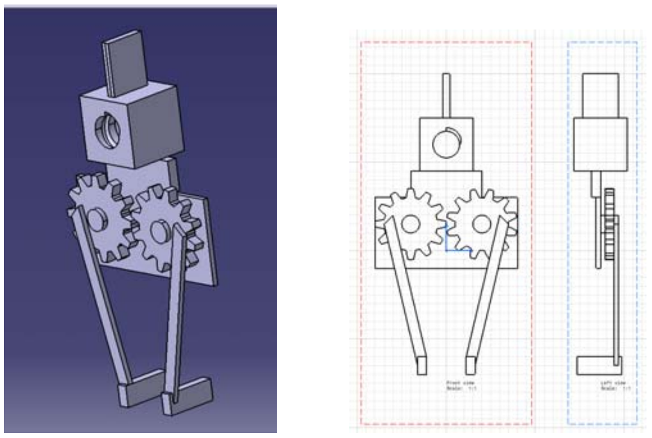


Fig 7. (a) 3-D view of the gripper. (b) Front and side view of the gripper.

Grabbing the chess piece, picking it up from the board and placing it back are similar kind of actions and can be grouped in a set. This set requires lowering of the gripper and holding the chess piece. Then it needs to be pulled up without disturbing the position of neighbouring chess pieces. This set of actions requires the movement of the grippers and joint 2.

The movement of joint 2 provides the lift necessary for picking and placing the pieces. The height of the lift should be such that while moving the grabbed pieces from one position to another, then it should be hit other pieces and thus should not disturb their position. This can be ensured by estimating the position from where the chess piece desired to be moved is grabbed by the gripper and the average height of the chess pieces above which this particular grabbed chess piece will pass during while traversing its path from its initial to the final position across the board.

For gripping the chess pieces spur gears are used on which gripping jaws are mounted. Actuation of the gears leads to the gripping action. The angle to which this gripping jaws can spread is decided by the angle by which this actuating spur gears (and in turn the shaft of the motor driving the spur gear arrangement) turn which is constrained by the distance r_L of the piece from the neighbouring chess pieces.

Moving the gripper to the initial position of the chess piece and taking it to the desired destination is another set. This set of task requires knowledge of the coordinates of the place where the chess piece is positioned on the chess board in reference to suitable axes.

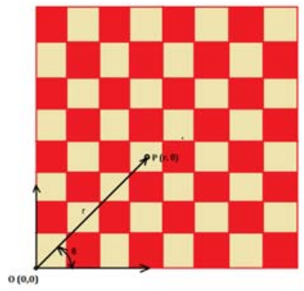


Fig 8. The polar coordinate system which is used to determine the location of the chess squares.

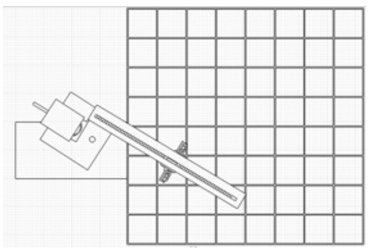


Fig 9. Movement of the manipulator over the chessboard (Not to scale).

If the manipulator starts with the end effector located at O (0,0) for reaching destination with coordinates P(r, θ), then the Joint 1 will move by angle θ and then joint 3 will rotate by a certain degree so that the end effector travels the radial distance 'r'.

VI. Robot Feedback and Control

As the name suggests, robot feedback refers to the sensory data from every joint of the manipulator which is used to effectively control the movement and orientation of the manipulator. The interface between the robot and the computer is via a control circuit with an AVR microcontroller at its centre. The microcontroller connects to the computer using the serial port via an USB to serial port converter. The manipulator motors are driven using high power motor drivers.

A. Sensory Data

Various sensors have been used to determine the instantaneous position of the manipulator. A 1 turn, 2 KΩ potentiometer was used to determine the angle of rotation of the manipulator about its base axis. It is referenced with a potential of 5 V and the value from the potentiometer is read in through the ADC with a resolution of 10 bits sampled at 172.8 kHz.

Two optical encoder wheels with IR sensors were used to determine the linear movement of the vertical and the horizontal screw drive mechanisms. The optical encoder wheel has ten protruding teeth with an IR sensor placed around it. So, for a count of 10 by the IR sensor, the screw has a complete rotation about its axis and hence moves the distance equal to the pitch of the screw. The minimum distance that can be determined by this movement is 1/10th of the pitch of the screw.

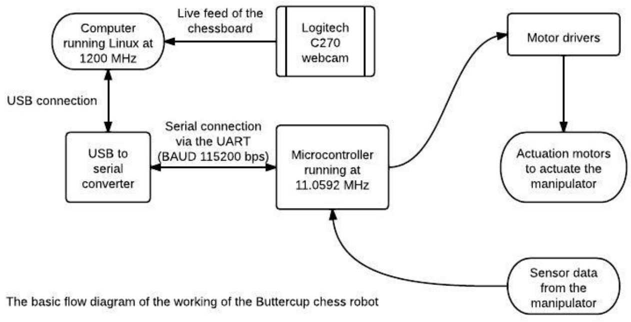

Computer running Linux at 1200 MHz
Live feed of the chessboard
Logitech C270 webcam
Motor drivers
USB connection
Serial connection via the UART (BAUD 115200 bps)
USB to serial converter
Microcontroller running at 11.0592 MHz
Actuation motors to actuate the manipulator
Sensor data from the manipulator
The basic flow diagram of the working of the Buttercup chess robot

Fig 10. Flow diagram of the working of the Chess Robot.

B. Control Circuit

The microcontroller, sensors, motor drivers and the serial port driver are the main elements of the control circuit. The microcontroller is an ATMEL ATMega32 microcontroller running at 11.0592 MHz. The output from the potentiometer (the rotation sensor) was input into the ADC2 pin and the output from the horizontal screw mechanism encoder was first compared using a comparator (LM324) and then input to the microcontroller. A 20 × 4 LCD (HD44780 driver) was connected to the microcontroller for debugging purposes. The motor drivers were built using complementary MOSFET transistors (IRF540 and IRF9540) and controlled using the microcontroller pins. The serial port driver was made using Hex Schmitt inverters (74HCT14) for Rx and Tx (UART) and an USB to serial port adapter. Everything was soldered on Vero boards. A 220 V AC to 12 V 10 A DC SMPS was used to effectively regulate the voltage output provided to the control circuit.

The computer connects to the robot through a serial connection to the microcontroller. The serial connection has a baud rate of 115.2 kbps. The manipulator is controlled through a set of commands which controls all the movements of the actuation motors precisely.

C. Chess Piece Movement using the Manipulator

The locations of the various chess squares (in algebraic coordinate notation) on the chessboard are calculated, and then stored in a table in the program which communicates with the robot via the serial interface. When the output from the GNU Chess engine is directly fed to the program, it then sends a series of commands to the microcontroller specifying the angle of rotation and the position of the screws for the source and the destination. The manipulator then picks up the desired piece from the source location and then places it to its respective destination position.

VII. Experimental Evaluation

The detection of the corners of the chessboard was done effectively using image processing methods. Challenges were faced when there were bad lighting conditions and skewed orientation of the chessboard under the webcam. It was found that the best image processing results were attained when the chessboard was oriented in parallel to the viewing angle of the camera. Choosing a threshold value of 50 to identify the color of the piece seemed to work in almost 99% of the cases where the ambient light came from an overhead white fluorescent light. After every move the opponent makes, the program is able to determine the movement of the piece between the old and new position. Errors in detection occurred when the lighting conditions were changed without altering the computation values. In dark lighting conditions, the algorithm detected false chess pieces at the edges and certain other portions of the chessboard. The GNU Chess Interface module sends and receives the moves by running GNU Chess in background and using its standard input and output stream to effectively communicate with it and extract the moves given by the engine. The engine returned „valid well thought out" moves in response to the user moves as determined by the image processing module. The interfacing and communication of the chess engine with the image processing module was perfect.

The gripper slipped a few times while gripping the chess piece. More padding and provision of optimum curvature to the gripping claws reduces the slippage. A more articulate gripper can solve this issue efficiently. The overshoot of the end effector of the manipulator is observed while it is about to arrive at the destination. With a trial and error method it was found that when the manipulator stops after execution of the commands (rotation of the joints) to perform the decided move, the revolute joint should be rotated for a small angle (most of the times less than 2 degrees) in the opposite direction (to which it was initially moving) to arrive at the destination properly.

VIII. References

[1] G.D.Illeperuma, Unversity of Colombo. "Using Image Processing Techniques to Automate Chess Game Recording" Proceedings, 27(2011) 76-83

[2] Cynthia Matuszek, Brain Mayton, Roberto Aimi, Marc Peter Deisenroth "Gambit : A Robust chess-Playing Robotics System", IEEE Int. Conf. on Robotics and Automation, May 2011, pp. 4291-4297

[3] Sokic E.; Ahic-Djokic M.; "Simple Computer Vision System for Chess Playing Robot Manipulator as a Project-based Learning Example", IEEE International Symposium on Signal Processing and Information Technology,Dec. 2008, pp. 75-79 .

[4] Learning Open CV Computer Vision with Open CV Library - Gary Bradski and Adrian Kaehler

[5] Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing".

[6] Daniel Molkentin, "The book of QT4 : the art of building Qt applications".

[7] "Designing Embedded Hardware" - John Catsoulis.