Chess Piece Movement Detection and Tracking, A Vision System Framework for Autonomous Chess Playing Robot

Dennis Aprilla Christie*, Tubagus Maulana Kusuma†, and Purnawarman Musa‡
*Faculty of Industrial Technology, †‡Faculty of Computer Science.
*†‡Gunadarma University, Indonesia.
Email: *dennis.ap.ch@gmail.com, †mkusuma@staff.gunadarma.ac.id, ‡p_musa@staff.gunadarma.ac.id

***Abstract*—Chess-playing robot is robots that is able to see, think, and move the chess piece properly. We separated the work into three parts (mechanics, engine, and vision), in order to ensure each of the parts work robustly. This paper is one of the series of those parts and will only focus on vision system for autonomous chess playing robot. Chessboard and chess piece detection is not a trivial task due to material and design diversity. A good chess-playing robot is able to see the chessboard and chess piece existence, and able to track chess piece movement generally and accurately. We propose a framework of chess piece movement detection algorithm. In general, there are chessboard detection, chess cell detection, chess piece movement detection, and notation creation. Contour Tracing is performed to select the region of interest, the chessboard, automatically, then Hough Transformation performed to identify the cells, then chess piece movement determination is perform by separating the Movement Detection into Stable and Unstable Frames Detection and Chess Piece Movement Detection. Finally the movement is converted into standard output, Portable Game Notation.**

***Index Terms*—Chess, Robot, Chess Playing Robot, Movement Detection**

I. INTRODUCTION

A. Background

Chess-playing robot is a robot whose goal is dedicated to chess, one of sports branches. It is highly necessary for chess-playing robot to have a good algorithm so it can act like humans when it plays. One of the initial construction Chess-playing robot is Turk, which was made in 1976 by Wolfgang von Kompelen that actually driven by human beings in the robot. And in 1997, for the first time in history, a human, of chess a master Garry Kasparov, was defeated by a of chess-playing robot called DeepBlue developed by IBM [2]. The event brought a new trend in this modern era, namely the development of chess-playing robot. The rapid development of Artificial Intelligence supports this new trend, and finally a competition for some researchers to make the best chess-playing robot arises, not only against a human, but against each other, a chess-playing robot.

Chess-playing robot is robots that is able to see, think, and move the chess piece properly. Urting et al. [2] create a chess-playing robots namely MarineBlue. They divide the chess playing robot architecture into three main parts: Chess-Engine, for the brain robots, Robot Control for the robots movement mechanics, and the vision for the robots sight. To make a complete functioning chess playing robot, those three parts has to be implemented properly. Therefore, we separated the work into three parts (mechanics, engine, and vision), in order to ensure each of the parts work robustly. This paper is one of the series of those parts and will only focus on vision system for autonomous chess playing robot.

Problem arise when the robot is dedicated to playing chess in general. General means, it suppose to work in various chess board and chess piece design. Chessboard and chess piece detection sometimes is not a trivial task to do. This is due to material and design diversity, it complicates vision system to recognize the environment. Various work has been conducted [1]–[5], however they also share different kind of methods and results. This is due to vision system for chess playing robot is still interesting research field to perform.

A good chess-playing robot is able to see the chessboard and chess piece existence, and able to track chess piece movement accurately. To overcome problem that is stated, we propose a framework of chess piece movement detection algorithm.

B. Problem Statements

Considered problems in this work are:

1) How chess-playing robot be able to recognize the entire chess pieces movements?
2) How vision system be able to identify the chess pieces and their movements?
3) How vision system can communicate with the robot so robot would able to perform the movement for the chess pieces?

C. Objectives

The aims of the work are:

1) Designing a vision system for the chess-playing robot to recognize the entire chess pieces movements.
2) Designing a tracking mechanism in every chess pieces movement.
3) Providing chess pieces movement code generation in PGN format that will be used as input for the robot so it able to perform the movement for the chess pieces.

D. Scopes

Vision system in this work assumes that the chess game runs normally without interference from the environment outside of the game, from beginning to the end.

II. RELATED WORKS

Several studies have been carried been carried out and each of them has unique methods. Nevertheless, in general, they shared similar method for chess piece movement detection.

Sokic et al. [1] scaled images acquired by camera into 240x240 resolution. They select ROI by using GUI and divide cells with numerical integration then Sobel operator used to detect the chess piece movement. They also build a triggering device for role timing decision between each player.

Urting and Berbers [2] divided their system's functionality into 3 layers; *pixel classification* to classify color of the pixel, *board layer* to detect chessboard and chess cells, *chessboard layer* to detect chess piece movement.

Banerjee et al. [3] performed their algorithm in dark red and lemon yelow color chessboard. Similar to Sokic [1] , they scaled image acquired by the camera but with 640x480 resolution. They select the ROI manually and detect cell with corner detection. To detect chess piece movement, they simple compare chessboard matrix between some temporal time.

Prakoso [4] perform their algorithm by using flat chess piece design with blue and red color to distinguish with the opponent. He used built-in function in OpenCV to perform chess cells detection.

Goncalves et al [5] calibrated their camera to find its pose relative to the chessboard and divide call with mathematical calculations then perform color classification. Chess piece movement detection performed by examining the existence of chess piece on the cell.

III. METHODOLOGY

A. Chessboard Detection

Urting et. al. [2] used corner detection by looking for the most extreme corner point and shows the automation process nicely. However when there is interference from outside environment, corner of some other object might detected and cause ROI adjustment not properly performed. Therefore, contour tracing performed instead of corner detection. The main idea is the contour which has largest rectangular shape is considered as a chessboard.

1) Preprocessing: Conversion to gray scale required by Canny edge detection. Gaussian filter in gray scale image used to reduce noise is necessary to perform before Canny edge detection [7]. To ensure the chess board edge are closely loop (contour), closing morphology performed to reconnect the disconnecting edges.

2) Approximation and True Chessboard: Contour tracing by Suzuki [8] divide the detected contours in topological structure. Outermost contour represented as a node with higher level in the topology. This topology is useful to decide the chessboard structure. Rectangular shape is the only shape that is acceptable, to verify only rectangular shapes contour is selected, approximation polygonal curve using Ramer-Douglas-Peucker (RDP) algorithm [9] performed. RDP algorithm returned a qualified points that is vertex of shape, the numbers of those qualified points can determine whether a contour is a rectangle or not. Largest area with image moment [11] constrain would be added together with shape and topology structure constrain to increase the confidence level of chessboard detection.

However, taking a decision that the chessboard has been detected by only one measurement is vulnerable to an error, especially in lack or over lighting condition. Shadow changes interference from external environment also cause false detection. To reduce occurrences of error, mode operation performed in a collection of chessboard approximation vertices.

3) Chessboard Wrap: Chessboard might not well positioned. Tilted chessboard due to user error, or trapezoidal chessboard due to camera perspective can complicate chess cell detection later in the series of algorithm. Therefore, geometric transformation needed to wrap chessboard into square shape. Each pixel applied by calculated 2D transformation matrix complete with interpolation. Pixels are finite, without interpolation there will be blank pixel in the transformed space.

B. Chess cell Detection

Chess cell detection searches the entire cells in the Chessboard, 8x8 cells in total. The goal of chess cell detection is giving an identity to each cells. Banerjee et al. [3] perform the chess cell detection by looking for an actual and probable corner. While the actual corner is a corner that they found using a corner detector algorithm [12], probable corner is a corner that they found by divide mathematically the extreme corner points into eight parts.

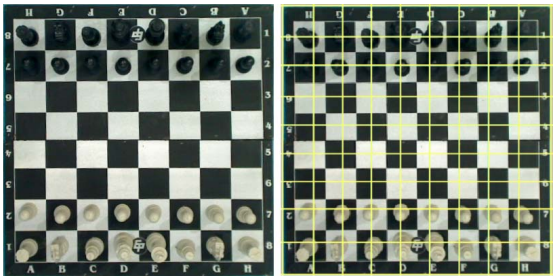


Fig. 1: Chessboard with border.

Similarly, Goncalves et al. [5] also mathematically divide cell into 8x8 cell. These methods provided more accurate result, but the fact that some chessboard design contains border (see Fig. 1), makes it seems it less general way to perform. Thus, we propose another method to perform chess cell detection.

1) Preprocessing (See Fig. 10a & 10b): It is similar to chessboard detection preprocessing, the only different is there is threshold performed after Gaussian filtering. Threshold is used to determine cells in chessboard since they commonly consist of bright and dark color (it is not always white and black, it depends on chessboard design. In Banerjee et al. [3] cases, they use dark red and bright yellow).

2) Line Detection (See Fig. 10c): Chess cell detection performed using Hough Transformation line detection [13]. To reduce outliers, Hough transformation applied in preprocessed

wrapped image of chessboard that previously obtained. Therefore, we expect perfectly straight lines, vertically and horizontally, that represent borders of the chess cell. However, it is impossible to directly obtain the desired lines. Undesired lines generally occur even though hough transformation parameters are carefully selected due to noise and outliers. There are two types of undesired lines, existence of diagonal lines and two closely adjacent straight lines (See Fig. 2).

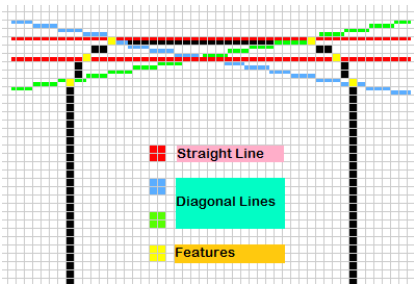


Fig. 2: Many Lines Detected by Hough Transformation.

3) Line Selection (See Fig. 10d): Diagonal lines obviously not line that form cell border in chessboard, thus these lines has to be removed, leaving only straight lines. Since Hough Transformation map all the feature from xy space into $\theta\rho$ space, it is straightforward to determine which line is slightly diagonal by checking its θ. Two closely adjacent straight lines have a different treatment, lowering threshold on Hough Transformation can't solve the problem, it is highly unstable. For this reason merging lines is chosen to resolve this issue. Merging defined as the formation of new lines which can represent both. With hough transformation it is straightforward to determine which line is closely adjacent to each other by checking its ρ. These processes guaranteed that only desired lines are left.

The number of desired lines are commonly 18 in total (9 vertical lines and horizontal lines). In some cases, it detected less than 18. This is due to chessboard border lines are sometimes not detected as a line. In this case, simple line addition to represent the chessboard border is conducted. 18 lines is needed for the next process.

4) Line Intersection (See Fig. 10e): Further, chess cell can be determined from the intersection between those lines. Intersection of two lines in xy plane, where lines L_1 represented as two points in space $(x_1, y_1), (x_2, y_2)$, the intersection point P between line L_1 and L_2 can be defined using determinants [6]. Determinants can be written out as equation 1. From 18 lines detected, 9x9 intersection exist, 81 points in total.

$$(\mathrm{P}_x,\mathrm{P}_y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}\right), \left(\frac{(x_1y_2 - y_1x_2)(y_3y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}\right) \quad (1)$$

In most cases, intersection points that collected do not have clear sequence order. Random intersection points complicate cell identification process. Thus, common sort algorithm conducted to overcome this problem.

5) Cell Identification (See Fig. 10f): Intersection points are still not able to represent a cell that can be understood by the system, the intersection points must be processed in such a way in order to make a correct components-forming cell. By knowing the intersection points mapping that contained in the chessboard (See Fig. 3a). According to Fig. 3c The components-forming cells can be formulated as equation 2.

$$\mathrm{C}_n = \{(\mathrm{P}_n, \mathrm{P}_{n+1}, \mathrm{P}_{n+9}, \mathrm{P}_{(n+9)+1}) | 1 \leq n \leq (81-9), n \neq 9, 18, 27, ...\} \quad (2)$$

Where C is cell, n is cell number, and P is corner coordinate point. Total number of point used are only 72 instead of 81, this is because the intersection points at the last row of the chessboard, has been represented by the points above it. Collection of components-forming cells are then stored for later process.

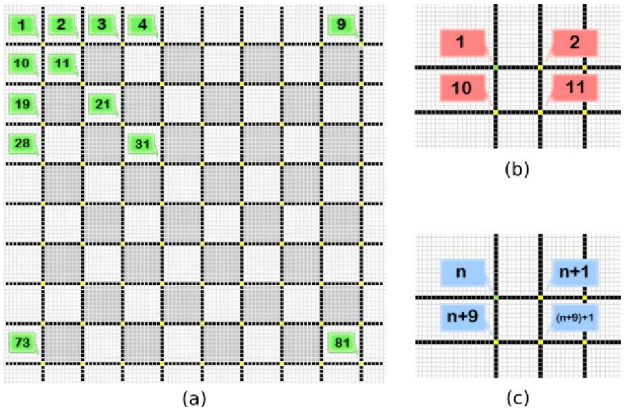


Fig. 3: Intersection Points in Chessboard.

C. Chess Piece Movement Detection

Sokic et. al. [1] build a triggering device, a replica of a chess clock, which is able to trigger snapshots by pressing a button or giving a command from PC right after chess pieces movement. Banerjee et. al. [3] after they perform an chess piece existence inspection on each chess cell, and the results will be entered into a predeclared matrix. Sokic method not fully automatic while Banerjee not mentioned how they detect the movement. Nevertheless, both methods gave nice results and inspired our work to detect chess piece movement. Thus, we create new method based on both methods combines.

1) Stable and Unstable Frame Detection: Movement detection can be performed by using image subtraction [14]. Problem arises considering chess piece movements occur concurrently with robot's arm or opponent's arm movement while displacing the chess piece. With only image subtraction, the system doesn't know whether chess piece has been moved. Therefore, the system must be able to separate the movement of the robot arm and the displacement of the chess piece. In other words the system has to know the exact two frames to compare chess piece displacement.

The idea of stable and unstable frames detection is considered that the situation when chess pieces movement performed by the robot-hand is an unstable situation. To make comparisons of chess pieces displacement, the situation must be said to be stable first.

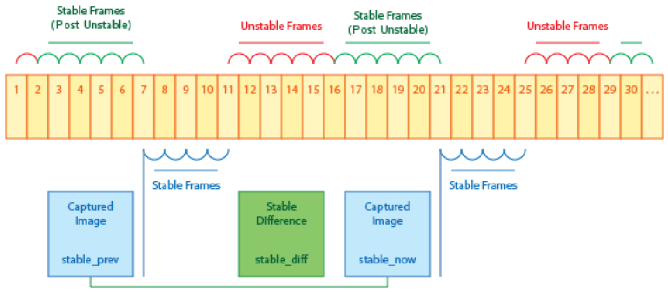


Fig. 4: Stable and Unstable Frames Description

Consider absolute image subtraction performed on the first frame I_1, and the second frame I_2. If the number of non-zero pixels from the subtraction results are below the threshold, then the situation said to be an approximation-stable. The process continues until I_n. If the number of approximation-stable has reached the predefined limit m, then I_m is an actual-stable, which will become reference frame to perform image subtraction for chess piece movement detection (See Fig. 4). Comparison of two frames that show the unstable situation is represented by red ◠. Comparison of two frames that show the approximation-stable situation represented by green ◠. Comparison of two frames that show the actual-stable is represented by blue ◠.

Stable and Unstable Frames Detection always produce images shortly after displacement was performed. Thus, the robot hand movement chess pieces is no longer a disruption to perform chess piece movement detection.

2) Blob Detection: Image subtraction again performed between two actual stable images. This operation results a white region of area that is called a blob. A blob represent an object that is moving in those particular frames, a chess piece movement (see Fig. 5). Some preprocessing and post processing necessary to do to create clearer blobs.

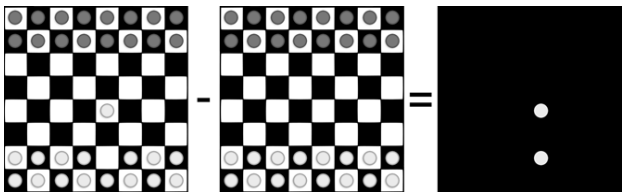


Fig. 5: Image Subtraction Between 2 Actual Stable Image.

3) Point Mapping: Position of blobs can be represented as their center of gravity which is obtained from their image moment [11]. Those points are then mapped into its corresponding cell in cell-space (see Fig. 6). The determination of which cell is corresponding to the points is done by checking points inside an area of a rectangle. Since component-forming cells are stored in two dimensional array, search function is implemented. Rather than brute force, row-column based searching is selected. Selected cells correspond to blobs position is called a working cell.

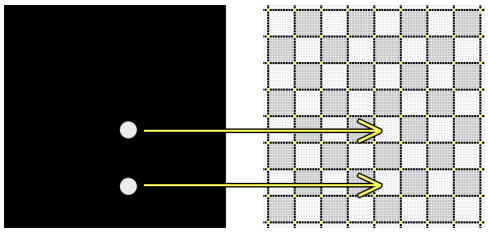


Fig. 6: Point Mapping.

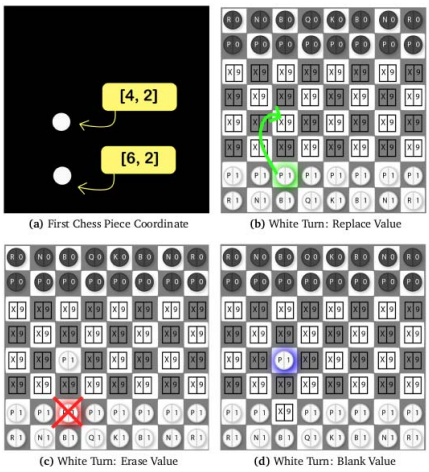
4) Matrix Update: Chessboard and chess piece data structure is created by using 8x8 two dimensional matrix and a customized struct contains chess piece identifier and color. First two rows and last two rows of matrix occupied by customized struct, represent initial state of chess board. Each time a chess piece movement occur, matrix values has to be updated. Based on the working cells, source and destination cells, displaced chess piece, and movement type, can be recognized. In general, there are four movements recognized by the system, normal moves, capturing moves, en-passant moves, and castling moves.

Normal moves is denoted by two active working cells. A cell that contains chess piece whose color is correspond to the player is a source cell, the other working cell is destination cell (see Fig. 7a). Capturing moves is similar to normal moves but with the other working cell is being occupied by opponent's chess piece. En-passant moves is denoted by three active working cells. As en-passant can only be performed by a pawn on its fifth rank, two horizontally parallel working cells is the source cell, while the other cell is the destination cell (see Fig. 7b). Castling moves is denoted by four active working cells. Castling moves can only be performed by a rook and a king and both of them had never move with cell among them is not being occupied. Since two chess piece involved during castling moves, there will be two pair of source-destination cells. Queen side castling determined if king and the left-side rook involved, while king side castling determined if the king and the right-side rook involved (see Fig. 7c).

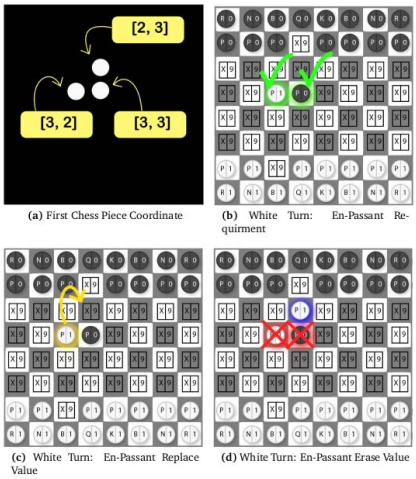
D. Notation Creation

Long algebraic notation is a standard method in Portable Game Notation (PGN) for recording ad describing moves among all chess game, books, and organization. Long algebraic notation text is our chess vision system final output. These texts are critical because it's the standard language for our vision system to communicate with brain system.

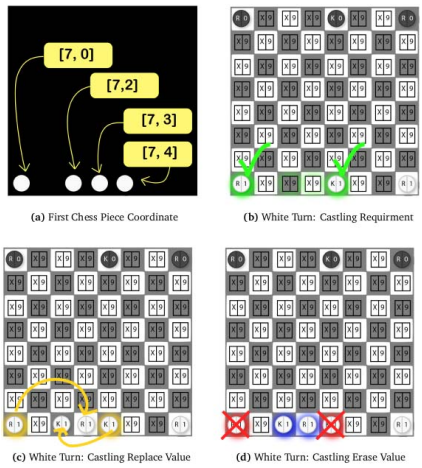
Notation creation needs 5 parameters, there are; the moving chess piece, its action (capturing or nothing), its source, its target, and its condition (check, en-passant, or nothing). Special treatment for castling moves, it only needs 2 parameters, there are castling types (king-side or queen-side) and its condition (check or nothing)



(a) Chessboard Matrix: White Turn.



(b) En-Passant Moves: White Turn.



(c) Castling Moves: White Turn.

Fig. 7: Chess Piece Movement Type.

IV. RESULT AND EVALUATIONS

To evaluate the performances of the algorithms, experiments are conducted on realistic chess board environment (see Fig. 8). All the developed software and conducted experiments are performed in a computer with Intel Dual Core P6200 processor, 2.13Ghz, 2GB Memory.

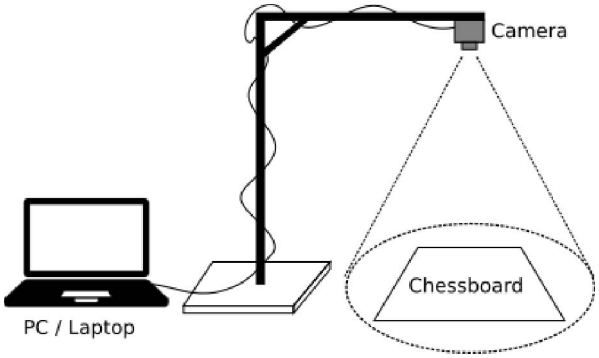


Fig. 8: Experiment Setup.

Chessboard detection using largest area, shape, and topology structure constrain gave nice but relatively satisfying result (see Fig. 9a–d). Due to noise caused by lack or over lighting condition, sometimes produce false detection. Collecting multiple chessboard detection measurement and conducted a mode operation increase higher confidence of chessboard detection.

Chess cell detection using Hough Transformation line detection in wrapped chessboard image is favourable to performed. It neglected outside environment, thus it effectively removes undesired outliers. Line detection, selection, and intersection produce a really nice result (see Fig. 10c–e). 72 cells are identified perfectly, indicated by 72 color rectangle shown in the image (see Fig. 10f).

Stable and unstable frames detection used to distinguish robot's or opponent's arm motion with chess piece displacement. Motion detection using absolute image subtraction result is shown in Fig 11. Motion is represented by white pixels in that particular image.

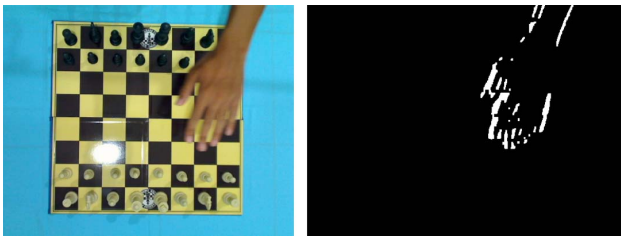


Fig. 11: Hand Motion Detection.

Since stable and unstable frames implemented nicely, movement detection using absolute image subtraction is trivial to perform. Remarkably, black chess pieces is harder to detect, especially in lack of lighting condition. It can be seen by comparing first image of Fig. 10a, and first image of Fig. 10b. Since black piece by nature has low contrast, sometimes it is fairly invisible by sight in the detection space. By adequate pre-process, this problem can be solved, the resulting blobs are still sufficient to be detected its coordinate in the image. (see last image of 10a and 10b).

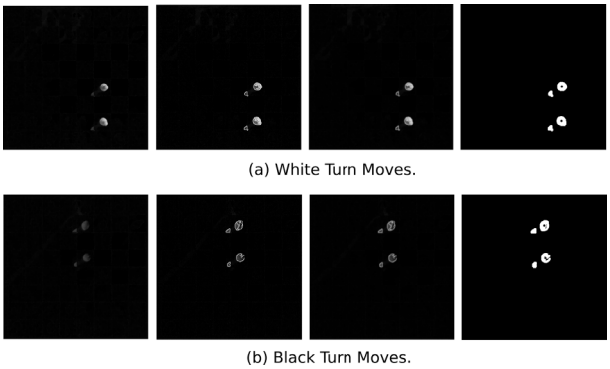
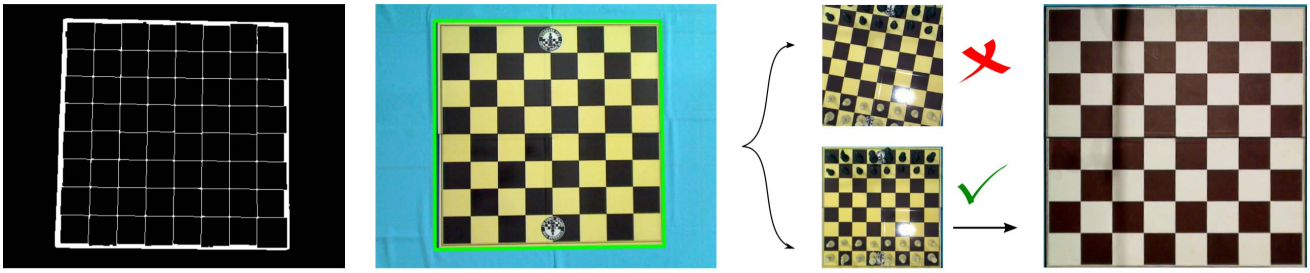
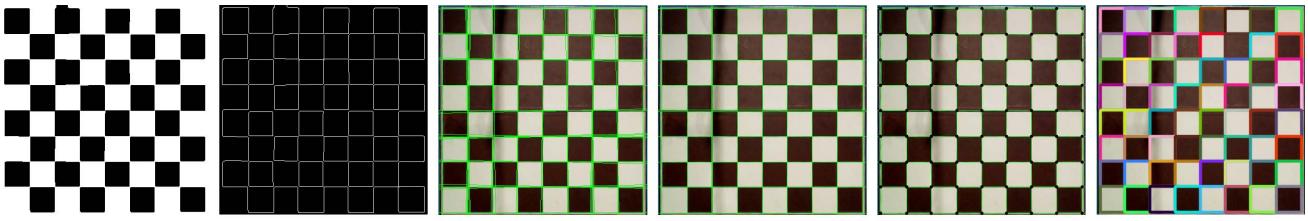


Fig. 12: Chess Piece Movement Detection.



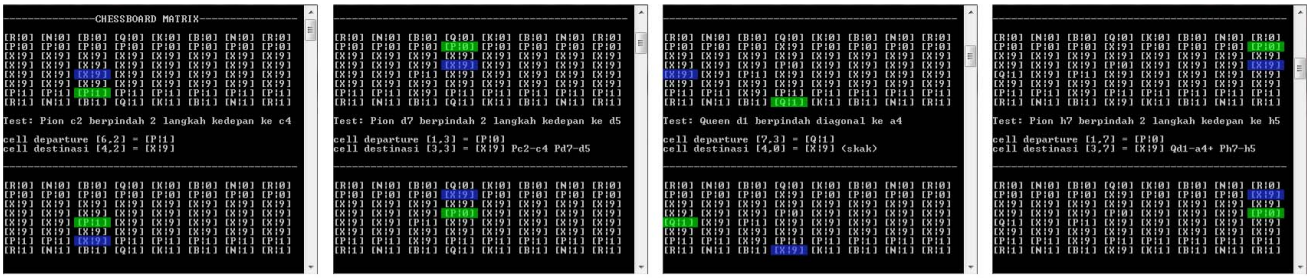
(a) (b) (c) (d)

Fig. 9: Chessboard Detection Results.



(a) (b) (c) (d) (e) (f)

Fig. 10: Chess Cell Detection Results.



(a) First White Turn. (b) First Black Turn. (c) Second White Turn. (d) Second Black Turn.

Fig. 13: Chessboard and Chess Piece Data Structure Update.

V. CONCLUSION AND FUTURE WORK

A vision system framework for autonomous chess playing robot has been proposed. Series experiments conducted to evaluate the performance of the framework in parts and overall system. All are working constantly and gives a satisfying result. System is still limited to promotion movement, therefore a chess piece recognition may improve framework performance. As this work is part of chess playing robot development, vision system will be integrated with engine and mechanics system.

VI. ACKNOWLEDGEMENT

This research was fully supported by Universitas Gunadarma, Jakarta, Indonesia. The authors gratefully acknowledge Universitas Gunadarma for providing research funding and for permission in using the research facilities.

REFERENCES

[1] Sokic, Emir, and Melita Ahic-Djokic. Simple computer vision system for chess playing robot manipulator as a project-based learning example. Signal Processing and Information Technology, 2008. ISSPIT 2008.

[2] Urting, D. and Berbers, Y. Marine blue: A low-cost chess robot. In Robotics and Applications. pages 7681. IASTED/ACTA Press.

[3] Banerjee, Nandan, et al. A Simple Autonomous Robotic Manipulator for Playing Chess Against Any Opponent in Real Time. Proceedings of the International Conference on Computational Vision and Robotics. 2011.

[4] Prakoso I. A. Rancang Bangun Robot Permainan Catur Berbasis Kamera. Masters thesis, Institut Teknologi Sepuluh Nopember. 2010. Surabaya

[5] Gonalves, Jos, Jos Lima, and Paulo Leitao. Chess robot system: A multi-disciplinary experience in automation. 9th Spanish Portuguese Congress On Electrical Engineering. 2005.

[6] Weisstein, Eric W. Line-line intersection. From MathWorld. A Wolfram Web Resource. 2002

[7] Canny, John. A computational approach to edge detection. Pattern analysis and machine intelligence 6 (1986): 679-698.

[8] Suzuki, S. and Abe, K. Topological structural analysis of digitized binary images by border following. In Computer Vision, Graphics, and Image Processing 30, pages 3246. 1985.

[9] Douglas, David H., and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization 10.2 (1973): 112-122.

[10] Shi, Jianbo. Good features to track. Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on. IEEE, 1994.

[11] Hu, Ming-Kuei. Visual pattern recognition by moment invariants. IRE transactions on information theory 8.2 (1962): 179-187.

[12] Shi, Jianbo. Good features to track. Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on. IEEE, 1994.

[13] Duda, Richard O., and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM 15.1 (1972): 11-15.

[14] Alavi, Shamir. Comparison of Some Motion Detection Methods in cases of Single and Multiple Moving Objects. International Journal of Image Processing 6.5 (2012): 389-396.