Chess Board Recognition

Raghuveer Kanchibail
School of Informatics and Computing
Indiana University
Email: rkanchib@indiana.edu

Supreeth Suryaprakash
School of Informatics and Computing
Indiana University
Email: skeragod@indiana.edu

Suhas Jagadish
School of Informatics and Computing
Indiana University
Email: jagadiss@iu.edu

***Abstract*—Chess Board recognition is an implementation which recognizes the chess board by locating the squares and detect the chess pieces from the input image using image processing techniques. The chess board is segmented from the input image, edges are detected using Canny's edge detector and cross lines are detected using Hough transform. This will give us the required 64 squares. Each square, with some vicinity around it, is extracted and compared to see if it contains a chess piece. If yes, then the test piece is scaled and oriented to compare with the pre-defined training set. The area score is calculated by taking difference of training pieces and test piece and the one with the lowest score is chosen as the best matching piece.**

Keywords: chess piece recognition, Canny, Hough, Area score

I. Introduction

Chess is strategic game played by two players on a chess-board which contains 64 squares. It can be a challenging game where the next best move can be tricky at times. This motivated us to build a solution which evaluates the current state of the game and predicts the next best move for the player.

Using image processing techniques, we can achieve this solution wherein the system detects a chess board(at any orientation), recognizes the different pieces and suggests the next best move. This can not only help humans to understand the game better but also can used by robots to play the game.

II. Image Processing

This phase involves detecting the given image, if it is a chess board or not and extract the required 64 squares from it. Basically we want to extract the chess board from the image which helps in detecting the cross lines more accurately and filter out the lines that are not the actual cross lines.

We start by converting the image to a grayscale image and then reduce it to a binary image. This way we segment the chess board and remove the background details. Edges are detected using Canny's edge detection followed by Hough transform to detect the straight lines. The steps involved to obtain the mask of the chess board are,
1) After Hough, each line is processed to obtain left, right, top and bottom mask. 2) Left mask is a binary image that has value '1' for all positions that lie to the right(below) of every line having positive slope. 3) Likewise right, top and bottom mask are calculated. 4) AND operation on these 4 masks will get the mask of the chess board.

We apply Otsu's binarization, Canny's edge detection and Hough's line detection again to the extracted chess board to eliminate the noisy lines.

III. Implementation

It was a challenging task to perform the above steps and we faced issues in various languages. We have implemented the code in C++, Python and Matlab.

We faced problems for Hough line detection with Python implementation wherein calculating the slope of the line was difficult with "HoughLine" function since it returned only rho and theta values. Even with "HoughLineP" function we faced few issues with minimum line length and maximum line gap. This was a bottleneck for us and hence we couldn't continue the implementation in Python. However below are the results achieved using Python[3][4][5] -

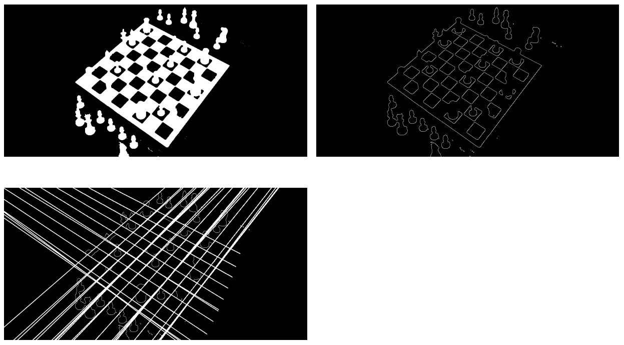
Fig. 1. Binary image after Otsu's representation, Image after Canny, Image after Hough

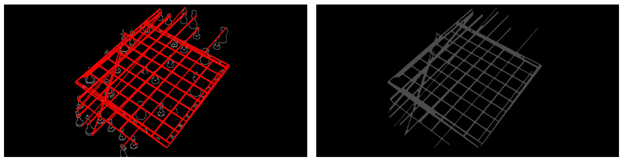
Fig. 2. C++ implementation - Hough Lines with chess pieces and result without chess pieces

We tried to implement the similar functionality using C++ with OpenCV package. We came to a phase wherein we could mask the chess pieces from the keyboard and erode the image to eliminate noises. But we struggled to continue with the piece detection phase and the results were not up to the mark. Fig 2 represents the results achieved using OpenCV implementation of C++.

We also tried implementing the algorithm using Matlab, using which we found more success. We integrated the Computer Vision System Toolbox with Matlab[7] and tried to implement the algorithm. We could achieve the results up to a phase where we segmented the chess board and retained only the chess pawns. As you will see below, the results are much better with Matlab and the programming was also simple -

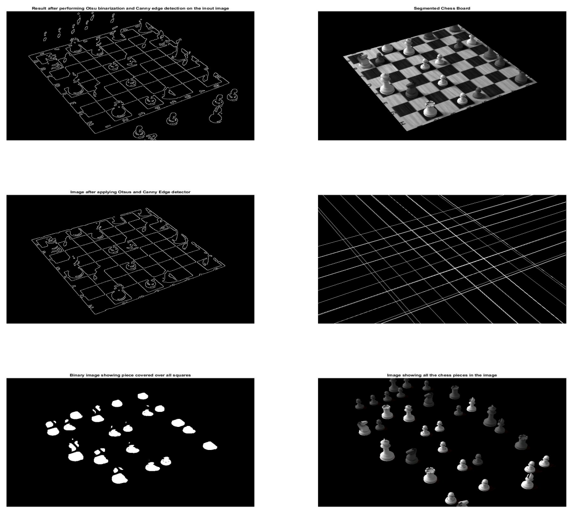
Result after performing Otsu binarization and Canny edge detection on the inout image
Segmented Chess Board
Image after applying Otsus and Canny Edge detector
Binary image showing piece covered over all squares
Image showing all the chess pieces in the image

Fig. 3. Images after each step using Matlab implementation

IV. Improvements

A. Chess piece detection

We had planned initially to extract each chess piece from the chessboard, orient it with respect to the training set and calculate the area score. But due to the challenges faced during implementations, we could not complete this task.

B. Orientation of the image

This algorithm will work for viewing angles between 30 to 60 degrees and will fail for other orientations. We need to come up with a better solution to recognize the chess board for all viewing angles.

C. Correctness of the game

For the detected pieces, we can check for correctness of the game by ensuring correct number of pawns are present on the board. For example, if there are 3 queens in the image, then the algorithm should give an invalid input message.

D. Intelligence to the algorithm

Once the pieces are recognized, intelligence can be added to predict the next best move of a particular piece.

V. Conclusion

This paper presents an algorithm to recognize the chess board using image processing techniques such as Otsu's binarization, Canny's edge detection and Hough transform, extract the chess pieces out of the board and measure the accuracy of the detected pieces. Our implementation involves coding in C++, Python and Matlab, results of which are represented in section 3.

Due to the implementation challenges faced as described in section 3, we could not complete the piece detection phase, however, we were able to understand the idea behind detecting chess pieces and we are planning to finish what we started(build an AI for predicting the next best move) during our summer break.

How to run the code

1. For Python - "python chess.py"
2. For C++ - "make main" and "./main image"
3. Matlab - Install matlab and integrate it with OpenCV using the reference[7]. Please note that Computer Vision System Toolbox must be installed while installing Matlab(it comes as a package). Install an appropriate c++ compiler (Visual studio 2012 preferred). Please refer to the documentation at[8]. Once the complete setup is done, run the file "Main.m" using Matlab tool.

Acknowledgment

We thank Dr. David Crandall for his support and guidance during the understanding and implementation of the project. We also want to thank the AIs for their continuous support. This course was really helpful in understanding the core concepts of vision and image processing techniques.

References

[1] Bhavani B.S, *Chess State Detection*, Stanford University

[2] Yangyang Yu, *Chinese Chess State Recognition*, Standford University

[3] Image Processing in OpenCV(Python), Image thresholding - *http://docs.opencv.org/3.1.0/d7/d4d/tutorial_py_thresholding.html#gsc.tab=0*

[4] Image Processing in OpenCV(Python), Canny edge detection - *http://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html#gsc.tab=0*

[5] Image Processing in OpenCV(Python), Hough line transform - *http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html*

[6] Koray, Can and Sumer,Emre , *A Computer Vision System for Chess Game Tracking*, Baskent University

[7] Matlab with OpenCV integration - *http://www.mathworks.com/discovery/matlab-opencv.html?refresh=true*

[8] C++ compiler setup for mex files in Matlab - *http://www.mathworks.com/help/matlab/matlab_external/choose-c-or-c-compilers.html*

[9] Hinterstoisser, S. ; Lepetit, V. ; Ilic, S. ; Fua, P., Dominant orientation templates for real-time detection of texture-less objects, Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference, pp 2257 2264, 13-18 June 2010

[10] Serge, Belongie; Jitendra, Malik; Jan, Puzicha, Shape Matching and Object Recognition Using Shape Contexts, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 24, April 2002