Chess Vision

Chua Huiyan

Le Vinh

Wong Lai Kuan

Outline

- Introduction
- Background Studies
- **2D Chess Vision**
  - **Real-time Board Detection**
  - **Extraction and Undistortion of Board**
  - **Board Configuration Recognition**
- **3D Chess Vision**
  - **Board Pre-Calibration**
  - **Extraction and Undistortion of Board**
  - **Board Configuration Recognition**
- Problems Encountered
- Conclusion
- Reference

Introduction

- Main Objective:
  - Real time recognition of perspective distorted chess board configuration.
- Our achievement:
  - Real-time recognition of the configuration of a 2D chess board that can be moved or rotated anytime.
  - Real-time recognition of the configuration of a 3D chess board that is pre-calibrated.

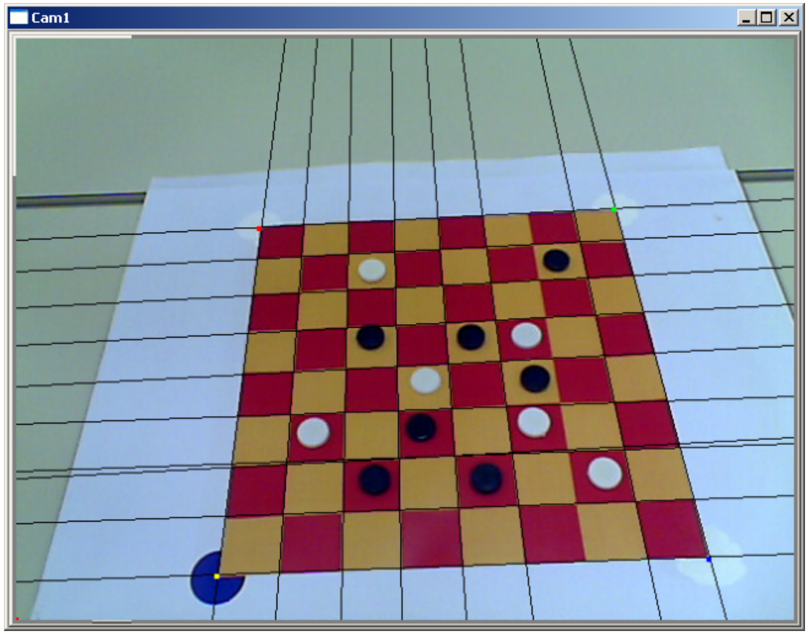
Previous Work

- To simplify the problem, previous chess vision algorithms [1, 2] have the following constraints / assumptions:
  - Camera is mounted directly on top of the board
    - Minimal perspective distortion.
  - Stationary chess board
    - Allow pre-calibration of chessboard.
  - Clean / plain background
    - Enable easy chessboard corner detection.
  - Known initial configuration
    - Configuration of the previous board configuration can be used to assist in determining the next board configuration.

Challenge of our project

- 2D chessboard recognition
  - Camera / board position and orientation can be changed in real-time.
    - Requires real-time tracking of chessboard corners and calibration of chessboard.
  - Unknown initial configuration
    - Allow any initial configuration that will be determined in real-time.
- 3D chessboard recognition
  - Camera mounted on a perspective view
    - Occlusion of chess pieces.

2D Chess Vision



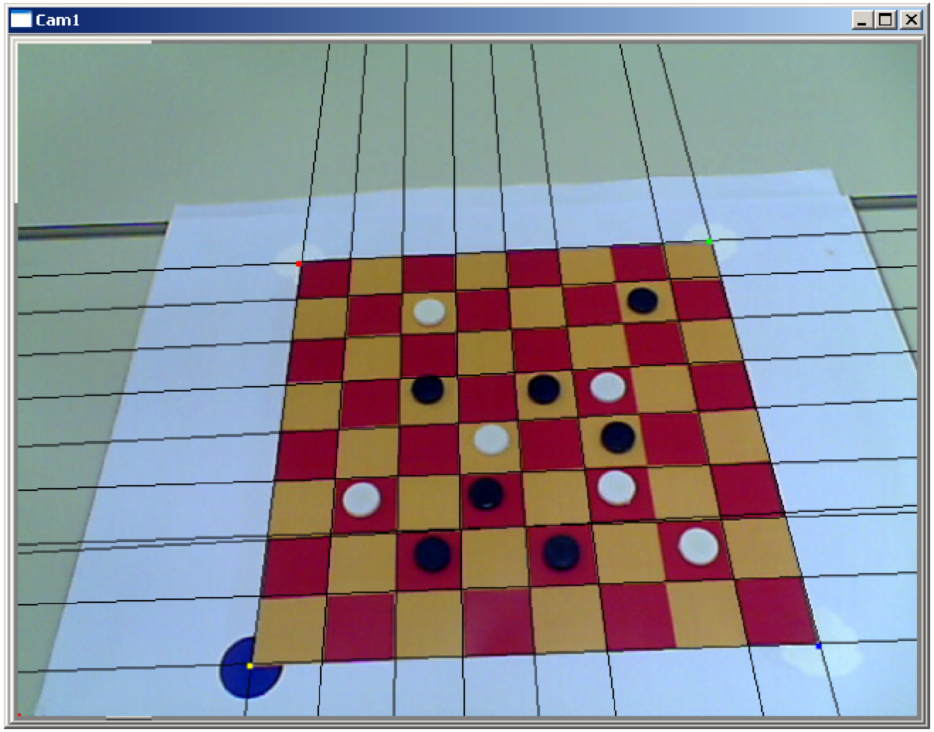
Step 1: Real-time Board Detection

1a. Board corners detection

- Combination of line detection and corner detection:

  Hough transform to detect lines and check for crosses with the detected corners to filter the outliners. Then 4 intersections by the borders are extracted.

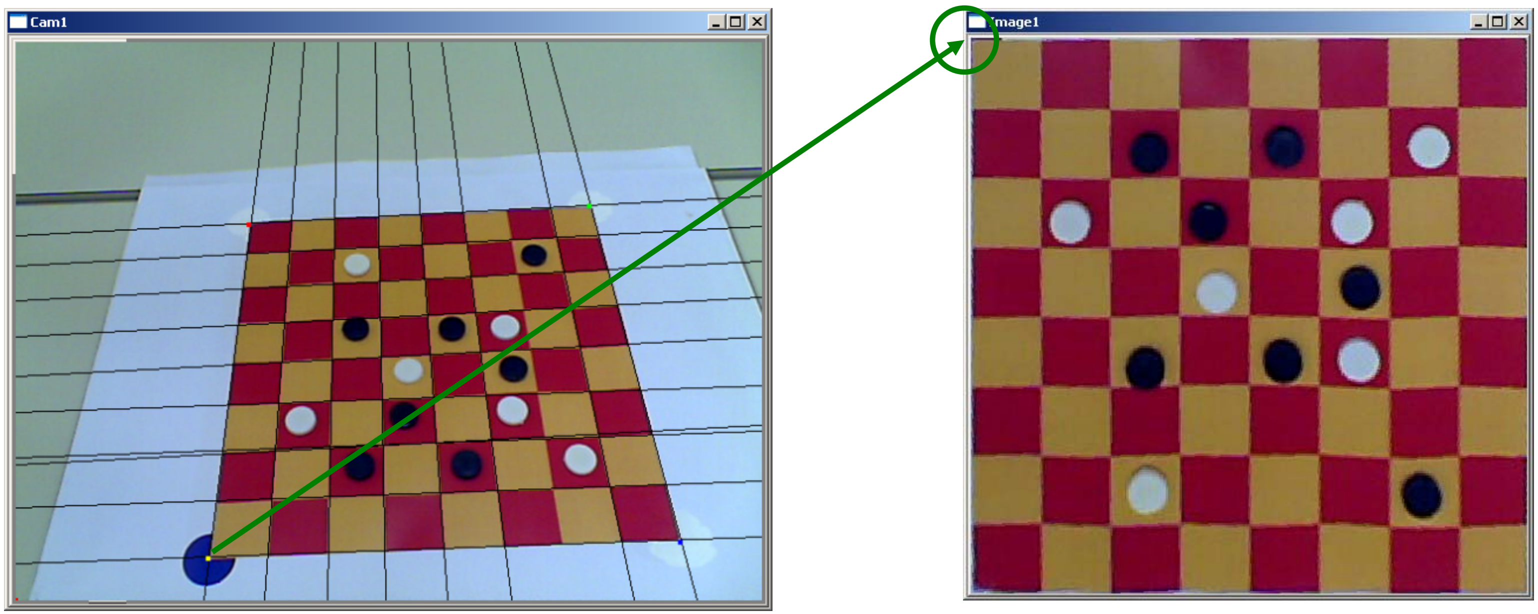
- This method minimizes the errors caused by noise and outliers but it's slower than other methods. However, the speed is adequate for our chess game context.



Step 1: Real-time Board Detection

1b. Board orientation detection

- We mark the top-left corner with blue color.
- After 4 corners are found, we detect the one with blue color, then sort them in clockwise sequence to send to next phase.



Step 2: Extraction & Undistortion of Board

- Using board corners detected from Step 1, extract and transform the board to a square board of size 480 x 480.
- This requires finding the perspective distortion, T of the captured board using the formula:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Destination scan was then used to undistort the image.

Step 3: Board Configuration Recognition

- Initially implemented method proposed by Farahat et al. [1].
- This method is very senstive to changing light condition.
  - Need to use a difference operator (between two consecutive image to compensate for lighting change) – even then it work best under lamp light.
  - We improved on this method to allow it to work on different lighting environment without using any difference operator or previous images.

Step 3: Board Configuration Recognition

3a. 1st Pass: Filter out non-occupied chess square

After getting the undistorted chessboard, Canny edge detection is applied to the whole undistorted image.

Divide the canny chess board image into 8 x 8 chess square images and apply threshold to detect whether a chess square is occupied.

Square without chess piece is represented as 0 in the system

Undistorted image

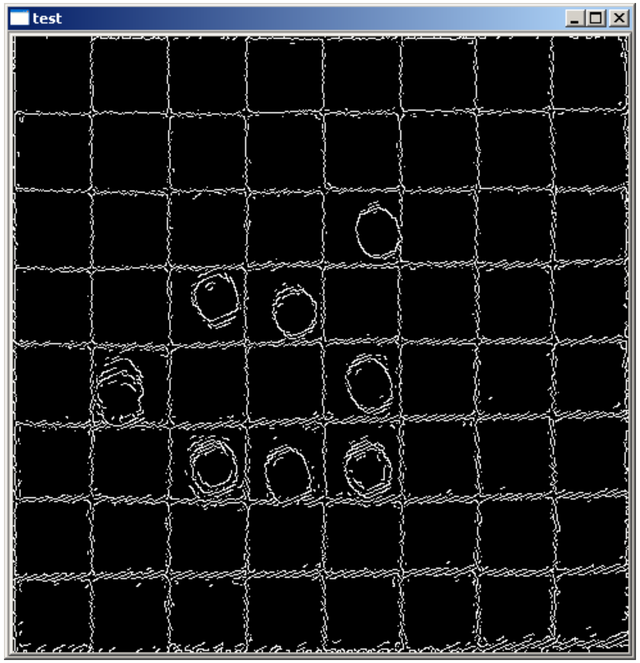


Image with Canny detection

Step 3: Board Configuration Recognition

3b. 2nd Pass: Determining color of chess piece in occupied chess square

- Image is first converted to HSV
- Value plane is used to determine whether the chess piece is black or white
- Pixels are classified into 256 bins in the histogram
- Black pixels are classified to range from the zero to the tenth bin
- Number of pixels found in the first 10 bins were summed up to track the number of black pixels in each chess square
- Chess piece is determined to be black (represented as 2) when the number of black pixels found in the chess square is above a threshold, else chess piece is white (represented as 1)

3D Chess Vision

3D Chess Vision

Step 1: Real-time Board Detection

- Same as 2D chess but it’s pre-calibrated.

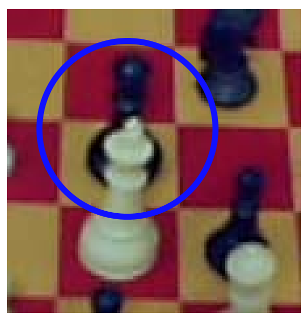
Step 2: Extraction & Undistortion of Board

- Same as 2D chess.

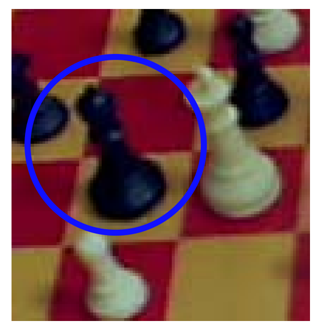
Step 3: Board Configuration Recognition

Setup

- Use two webcams positioned perpendicular to each other so that pieces appear occluded in one view may be seen from another view.
- Inititial configuration of board is provided.



1st view



2nd view

Step 3: Board Configuration Recognition

Step 3a: Determine the two chess square

- Divide the chess board image into 8 x 8 chess square images
- For each chess square
  - Obtain the abs difference images of both views for two consecutive frames.
  - Perform binary threshold – set difference value above 30 for each pixel to 1, otherwise 0.
  - Compute the total sum square difference for both difference image.
- Find the two chess squares with maximum total sum square difference.

Step 3: Board Configuration Recognition

Step 3b: Determine the changed configuration

- If the original states one of the selected chess square = 0 (unoccupied)
  - Swap the states of the two square
- Else (both squares are occupied)
  - Use Laplace to find edges of the current image for both squares.
  - Replace the state of chess square with more edges with the previous state of the chess square with less edges.
  - Set the state of the chess square with less edges to 0 (unoccupied).

Implementation & Testing

- Implementation:
  - C++, OpenCV, OpenGL for Vision part.
  - Java socket programming for interface with game engine.

- Testing:
  - Perform stress tests to test for worst case scenario.


  - Perform testing under changing light condition.

Problems encountered

- Some image analysis methods work well for static images but very unstable when implemented in real-time.

- Real-time corner detection
  - Trivial methods such as simple corner / color / corners detection is very unstable.
  - Solution: Use a combination of various methods to determine the corners.

- Real-time integration
  - Sensitive to change of lightings / flickering fluorescent light / reflection.
  - Solution: Iteratively change our methods to be more robust to changing environment conditions.

- Crashing
  - Caused by two threads trying to access a same image file.
  - Solution: Implement a semaphore for locking the files accessed by the two threads.

Conclusion

- What we achieve?
  - Concrete implementation of the game Lines of Action with chess vision and AI module (2D version).
  - Successfully implemented both the 2D and 3D chess recognition.
  - Improved on the robustness of the lighting
  - Camera do not need to be mounted directly on top of the board
  - Chess board can be moved around in the middle of game play in the 2D version without affecting chess recognition
  - 2D version can take in any input configuration
  - Background can allow for some noise

Conclusion

- What we have learnt?
  - Learnt to develop a computer vision system and implement in real-time.
  - Learnt to deal with increased noise in real time video due to change of light condition.
  - Applied theories that we have learnt in class: Canny edge detection, corner detection, Hough transform, homography, color spaces.

Reference

[1] A. K. Farahat, A. M. Hassan and M. A. El-Nagar, *A Vision System for Chess Playing Robots*, 46th IEEE Midwest Symposium On Circuits and Systems, December 27-30, 2003.

[2]David Urting, Yolande Berbers (2003), *MarineBlue: A Low-Cost Chess Robot*, Proceedings of the IASTED International Conference on Robotics and Applications (Hamza, M.H., ed.), pp. 76-81.

Thank you

Chua Huiyan

Le Vinh

Wong Lai Kuan