Paweł SUCHECKI, Marcin IWANOWSKI

Warsaw University of Technology

Draughts playing system with vision-based interface

Abstract. *In this paper the draughtsplaying system based on the vision user feedback, is presented. The system plays draughts with a human players. It consists of two sub-systems: vision-based human-computer interface and draughts-playing engine. Two variants of vision user interface are introduced: the first one with application of computer and camera, the second one with application of computer, camera and overhead projector. The draughts-playing engine is based on the minimax algorithm with position estimation and some modifications: alpha-beta pruning, transposition tables, iterative deepening, quiescence search and null window heuristics. The efficiency of the systems has been proved by two draughts matches (eight games): between the system and the Champions of Poland in classical draughts. The results of the games prove that the application based on the method introduced in this paper is not only offering easy-to-use smart user interface but is able to compete with the best human players.*

Streszczenie. *W niniejszym artykule prezentowany jest system grający w warcaby wykorzystujący do komunikacji z użytkownikiem wizyjne sprzężenie zwrotne. Przedstawione są dwa warianty wizyjnego interfejsu użytkownika - pierwszy do zastosowania z komputerem oraz kamerą, drugi do zastosowania z komputerem, kamerą oraz projektorem. Silnik gry w warcaby oparty jest na algorytmie minimax oraz jego usprawnieniach: odcinaniu alpha-beta, tablicach transpozycji, pogłębianiu iteracyjnym, przeszukiwaniu stanów stabilnych oraz pustym oknie przeszukiwań. O efektywności systemu świadczą wyniki dwóch meczów warcabowych (osiem partii): pomiędzy systemem a Mistrzami Polski w warcabach klasycznych. Rezultaty partii dowodzą, że aplikacja zrealizowana w oparciu o przedstawioną w niniejszym artykule metodę nie tylko oferuje łatwy w użyciu inteligentny interfejs użytkownika ale także jest w stanie rywalizować z najlepszymi zawodnikami.* ***(System grający w warcaby z wizyjnym interfejsem użytkownika)***

Keywords: game playing systems, computer vision
Słowa kluczowe: systemy grające, widzenie komputerowe

Introduction

In this paper the draughtsplaying system based on the vision user feedback is presented. The system plays draughts with a human players. It consists of two sub-systems. The vision subsystem allows communicaiting with the computer using pointer and gestures. The game-playing subsystem – draughts-playing engine is responsible for finding the optimal move of the computer-player at each level of the game. Two variants of vision-based user interface are proposed: the first one with application of computer and camera, the second one with application of computer, camera and overhead projector.

The paper consists of three parts. The first part concerns the visual sub-system. Important concepts of the interaction between user and system and the scheme of the vision feedback loop are introduced. Image processing techniques: morphological filtering and two variants of image segmentation – by binarization and by using temporal image variations, are also described.

The second part of the paper concerns the construction of the draughts playing engine based on the minimax algorithm. The method of draughts position estimation is introduced. The generic algorithm of the game tree searching minimax is described as well as its' modifications: alpha-beta pruning, transposition tables, iterative deepening, quiescence search and null window heuristics.

The third part of the paper concerns the application implementing the developed method and the results of its execution, while the last part concludes the paper.

Vision-based interface

Concepts of the interface

Two variants of vision-based user interface are introduced. In both variants the main object of an application window is the draughts-board, that presents the actual state of the game. Player executes the moves using the pointer, that is automatically detected by the vision sub-system. In the first variant – with no application of overhead projector – the camera is directly pointed at the user and continually acquires the images (Fig. 1). Every image is overlayed by the half transparent image of a draughts-board, before it is displayed on the screen. This way, user can see the board as well as the position of a pointer. When intersection of the pointer and the field of the board lasts for a particular time, it is assumed that the user's intention is to select that field. The execution of move consists of selection the field, on which the draughts-piece is located, and then selection the field, on which the piece should be moved.

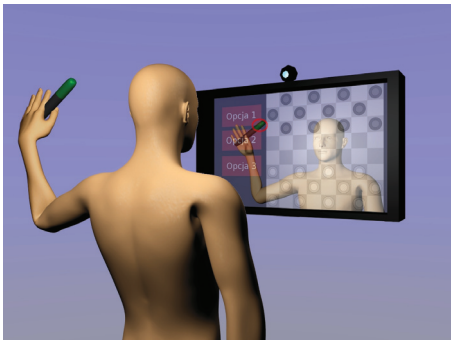

Fig. 1. Model of vision-based interface, first variant – without projector.

The second variant of vision-based user interface is designed for use with the overhead projector. The application window is displayed at the projection screen. Player uses laser pointer to select the fields of the board and execute the moves (Fig. 2).

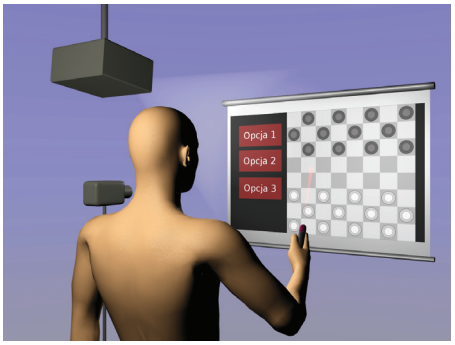

Fig. 2. Model of vision-based interface, second variant – with projector.

Computer vision methods

In the first variant of user interface, the automatic detection of the pointer is based on the assumption, that its' colour clearly distinguish from the background. The segmentation of the colour image acquired by camera goes as follows: At first, image is decomposed into three channels: red, green

and blue. Then every channel is binarized by thresholding. The values between the upper and the lower threshold correspond to the pointer's values of pixels for each channel. The intersection of these three binary images forms the binary image, that represents the pointer.

In the second variant of user interface the segmentation is based on the variation of images in time. In this case the camera is pointed at the projection screen and observes the application window, that remains static over the particular period. The time-varying component of the scene is the laser spot, that is used by the human player to move across the display area. By comparison of two successively acquired images, one gets the position of the laser spot in two consecutive moments of time.

In both variants of user interface the morphological filtering [1, 2, 4] is performed to remove noise from binary image. Morphological opening – erosion followed by dilation – allows to remove small objects from the foreground of an image and thus to better isolate the pointer object.

The camera calibration

The camera calibration is necessary in the second variant of user interface. Its purpose is to create a map identifying between pixels in the image registered by camera and fields of the board in image displayed on the projection screen. This operations is accomplished in few steps (Fig. 3):

Fig. 3. Camera calibration.

First, the draughts board image is displayed and it is registered by camera. Then, the null image is displayed and again the camera aquires the image. The registered images are substracted. The received image is binarized and then labeled. Each label refers to a individual field of the board.

The algorithm of vision-based feedback

The simplified schema of vision-based feedback (without verifying the correctness of player's moves) goes as follow:

1. *display* image in application window
2. *acquire* image by camera
3. *detect* the pointer in the image
4. *identify* the field of the board corresponding to pointer's position
5. *if* the same field is not permanently identified over the particular period, *then goto* 1
6. *if* none of fields has been selected *then select* the identified one and *goto* 1;
   else register move from previosuly selected field to the just indicated one
7. *if* there is no possibility of making another move *then stop* – player wins the game;
   else compute and execute the system's move
8. *if* there is no possibility of making another move *then stop* – system wins the game
9. *goto* 1

Draughts playing engine

The second sub-system is responsible for executing the system's move based on draughts position estimation.

Min-max algorithm

A minimax [5] is a basic algorithm for choosing the next move in a two-player zero-sum game of perfect information. In draughts, players make alternate moves. For every position that can be reached as a result of player's move, another set of positions can be reached as a result of his opponent's move and so on. This consideration can be illustrated as a game tree, where the nodes represent the positions and the branches represent the moves (Fig. 4). The task of a game

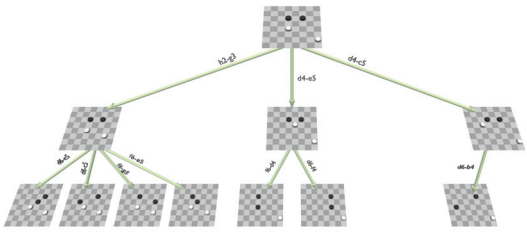

Fig. 4. The game tree.

tree searching algorithm is to find the move from given position, that leads to the best possible position in few moves, assuming, that the opponent also makes the best moves for himself.

The minimax algorithm accomplishes this task by assigning a value to every encountered position of a game tree of given depth. The value indicates how good it would be for a player to reach that position. The process of associating values with positions goes as follows. Every position at the final depth of the tree is evaluated. Then assigned values are propagated back to the root of the tree: At every level the value of the node is maximum of its' children values if whites' turn[1] or minimum if blacks' turn, as one player tries to maximize his own minimal gain and the other tries to minimize his own maximum loss (Fig 5). Finally, the move that gives the best score at the root level is chosen.

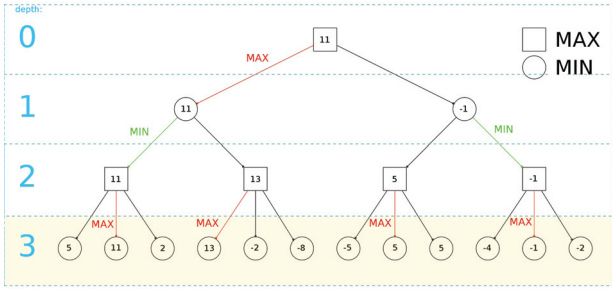

Fig. 5. Minimax – propagation of node values throughout the game tree.

Position evaluation function

The importance of position evaluation function for minimax algorithm is fundamental. The moves are chosen based on positions values and thus based on the evaluation function. The correct evaluation of draughts position is one of the hardest elements of the game. Even a small change in the position of a single piece can dramatically affect the situation on the board. However, there are some general rules, that can be applied to rough estimate draughts position.

The most important element of the position evaluation is a material situation on the board $E(m)$. Most often, the advantage of even only one piece is enough for victory. Material situation is computed using the following equation:

$$E(m) = (w_p - b_p) + 2.5(w_q - b_q), \tag{1}$$

[1]We assume here that computer plays whites, while its counterpart – blacks.

where w_p is the number of white pawns, b_p - number of black pawns, w_q - number of white queens, b_q - number of black queens.

Another basic factors affecting the position estimation are tempos balance t_b, the control over the centre fields of the board and the number of piece columns. The tempos balance indicates how much one player is ahead of another in developing position. It is calculated as follows: every white piece on the first line of the board earns one point, every white piece on the second line earns two points, and so on. Every black piece on the eighth (last) line of the board gets one point, every black piece on the seventh line gets two points, and so on. The balance is equal to the difference between the sums of points for white and for black side. Since it is better to have less tempos at the beginning of the game and more in the endgame, the appropriate coefficient is computed as:

$$E(t) = \begin{cases} -t_b & \text{in beginning phase of the game} \\ 0 & \text{in middle phase of the game} \\ t_b & \text{in endgame phase} \end{cases} \tag{2}$$

An elementary principle of draughts tactics is to control the centre fields of the board. These fields are c3, e3, d4, f4, c5, e5, d6, f6. Any pieces stationed on these fields should affect the evaluation, what is considered in the following maner:

$$E(c) = w_c - b_c, \tag{3}$$

where w_c and b_c are the numbers of white and black pieces in the centre, respectively.

Three or more pieces placed side by side in one direction form a column. Columns have positive impact on position as they affect its stability and create strategic threats:

$$E(g) = w_g - b_g, \tag{4}$$

where w_g and b_g are the numbers of white and black piece columns respectively.

The evaluation function returns the value indicating how good the position is for the players: the higher the value – the better position for whites, the lower the value - the better for blacks. Taking into consideration all above factors, the evaluation can be expressed as:

$$E(p) = k_m E(m) + k_t E(t) + k_c E(c) + k_g E(g), \tag{5}$$

where k_m, k_t k_c and k_g are the weights of factors to overall evaluation.

Alpha-Beta pruning

The pure minimax algorithm visiting all nodes of the game tree is inefficient. One of the major refinements for reducing the number of positions that has to be search is $\alpha - \beta$ pruning. The algorithm is based on observation, that some branches of the search tree can be eliminated without affecting the final result of computation [6]. Player will not have a chance to play a particular move if his opponent can avoid it by choosing the other, more adventageous variant for himself. Thus, further analysis of this particular move is pointless. The example of reducing the size of the tree by pruning branches is shown in figure 6. Having investigated the first move from position marked with the C letter, the analyse of remaining moves is avoided. The value of C node can only get higher (the move is executed by the player that maximize the score), so the player minimizing the score in node B would choose the move that leads to node A, as this is more

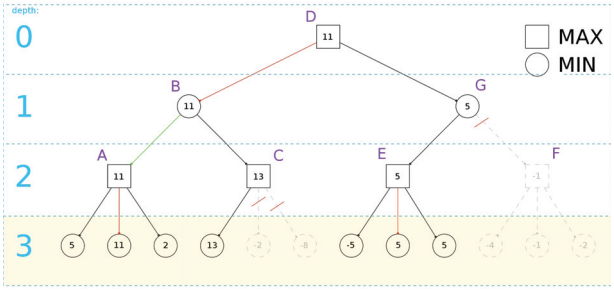

Fig. 6. Alpha-Beta pruning.

promising variant from his point of view. Similar situation occurs in the G node – after investigation of E node, it is known that the value of G node would be worse than B for the player that maximize the score.

There are two bounds in $\alpha-\beta$ algorithm: α – the highest value, that has been reached by the player maximizing the score, β – the lowest value, that has been reached by the player minimizing the score. Whenever the condition $\alpha \geq \beta$ occurs, the cut-off takes place.

Algorithm efficiency depends on the order of moves investigation. The largest number of prunings occurs when the best move is examined as the first one. In the worst case there is lack of prunings at all and all nodes are visited as in the minimax algorithm.

Other enhancements

In draughts, the same position can be reached in multiple ways. If the same position is encountered again, but from different sequence of moves, it is examined once more. To avoid redoing the entire search again, the mechanism of transposition tables is applied. After the position is analysed, the results are stored in transposition table (in practice implemented as large hash table). Searching the next position is preceded by verification whether the transposition table contains information about it. If obtained information can be used, the position does not have to be searched again. Even if the transposition table entry comes from the search of depth that is not big enough, it is still possible to use the best move from a previous search to improve the move ordering - as the best move from the perspective of few moves is a good candidate to also be the best in the context of analysis of greater depth.

Another enhancement of $\alpha - \beta$ algorithm focused on move ordering is iterative deepening. Instead of searching to a given depth immediately, it assumes repeatedly invoking a search routine with increasing depth until given depth is reached or time limit is exceeded. Apart from the benefit of time management strategy, it affects the efficiency of $\alpha-\beta$ algorithm as it enables the move ordering based on the results derived from previously performed shallower searches.

The game tree search to a fixed depth is exposed to a problem called 'horizon effect'. It manifests itself with situation, when the best move from fixed depth search turns out to be the wrong one from the perspective of deeper search (e.g. one can win a pawn in fifth move but loss the three ones in the sixth move). To minimize this problem, instead of stopping the search and evaluating position when the desired depth is reached, the routine called 'quiescence search' is invoked. It performs additional searching, ensuring that only the stable positions are evaluated. In draughts the position can be considered stable, when there is no possibility of making a capture move.

In minimax algorithm with $\alpha - \beta$ prunings, the range of

values between the α and β bounds forms a window search. The narrower the window, the more cut-offs occur. Many improvements of algorithm are based on the manipulation of this range. One of them – MTD(f) algorithm – uses only null window, where $\beta = \alpha + 1$. Searching with null window performs a boolean test whether a move produces a worse or better score than a passed value. By carrying out the series of tests, the true score of position and the best move is determined [3].

Implementation and results

Implementation

The system is implemented in C language. The implementation of vision user interface is based on OpenCV library. The positions on the draughts board are represented by the bitboards data structure. It allows to encode the position by three 64-bit numbers and use the bitwise operations to generate moves.

The main window of the application is shown in the figure 7. The draughts board and application options are the

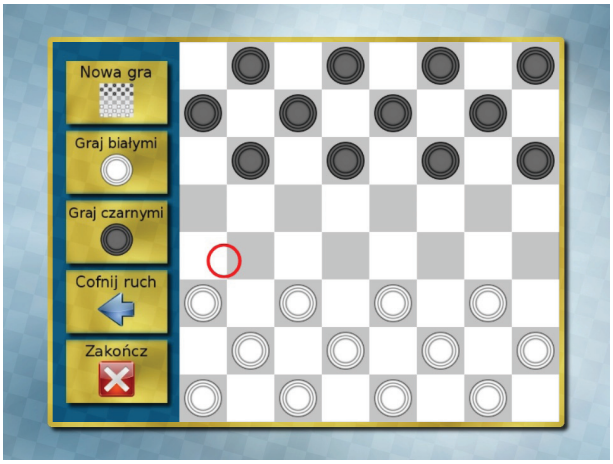

Fig. 7. The application window.

interactive elements of the interface. The current position of the pointer is indicated by red circle. Available options allows to start a new game, choose the colour of pieces, undo moves and quit the application.

The implemented application in both variants of user interface is shown in figures 8 and 9 respectively.

Fig. 8. Vision-based interface without projector – implementation.

Results

To test the strenght of game engine, two draughts matches has been organised: between the system and the Champions of Poland - grandmaster Marcin Grzesiak and master Michał Janicki. The matches has been played according to the official rules of Polish Draughts Federation.

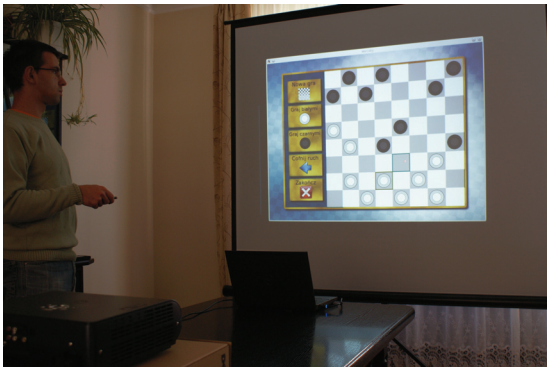

Fig. 9. Vision-based interface with projector – implementation.

The time control was 45 minutes with an addition of 30 seconds per move. Each match consisted of two rounds and each round consisted of two games. In every round the starting position of games was chosen randomly according to the „flying draughts" rule.

Both matches – and each game – ended with a draw result. In one game the system achieved winning position (in draughts theory known as Pietrov triangle), but couldn't play the endgame properly. Another enhancement of game engine – the endgame database – would provide system with perfect play in the endgames.

In the other games, the champions had the initiative but not enough for victory.

Conclusions

The draughts-playing system with vision-based human-computer interface was described in the paper. It allows comunicating with a computer by only using of markers and gestures. The detection of the intention of the human player is performed by analysis of a camera attached to the computer. Two versions of the vision interface was proposed: with and without overhead projector. The second part of a system is a draughts-playing engine, which is based on the minimax algorithm with position estimation and some modifications: alpha-beta pruning, transposition tables, iterative deepening, quiescence search and null window heuristics.

The efficiency of the system has been proved by two draughts matches: between the system and the Champions of Poland in classical draughts. The results of the games prove that the application based on the method introduced in this thesis is able to compete with the best draughts players.

BIBLIOGRAPHY

[1] Iwanowski M.: Metody morfologiczne w przetwarzaniu obrazów cyfrowych EXIT, Warszawa 2009
[2] Soille P.: Morphological image analysis Springer Verlag, 1999, 2004
[3] Plaat A., Shaeffer J., Pijls W., de Bruin A.: Best-first fixed-depth game-tree search in practice, proc.of IJCAI, Montreal 1995
[4] Serra J.: Image analysis and mathematical morphology, vol.1, *Academic Press*, 1983
[5] Shannon C.: Programming a computer for playing chess, *Philosophical Magazine*, 1950
[6] http://chessprogramming.wikispaces.com/Alpha-Beta

Authors: Paweł Suchecki, M.Sc.; Marcin Iwanowski, Ph.D. Institute of Control and Industrial Electronics, Faculty of Electrical Engineering, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warszawa, Poland, email: iwanowski@ee.pw.edu.pl