2nd International Conference on Autonomous Robots and Agents
December 13-15, 2004 Palmerston North, New Zealand

Autonomous Game Playing Robot

Donald G. Bailey, Ken A. Mercer, Colin Plaw
Institute of Information Sciences and Technology,
Massey University, Palmerston North, New Zealand
D.G.Bailey@massey.ac.nz

Abstract

A key objective for a game playing robot is to have the inferface between the player and the robot be dictated by the game rather than explicit computer requirements. The result is a robot that plays autonomously, without the need for conventional computer interfaces. The requirements for having a computer play board games in the real world include: a suitable game, including playing pieces; a vision or similar input system to sense the game state; a manipulator or similar output system to change the game state; and a controller that coordinates the interaction between the human and the computer. Issues discussed are lighting, camera calibration including correcting for distortion, resolution of both sensor and manipulator, coordinate systems and conversion, and user interaction issues such as negotiating the roles of each player, handling invalid or incomplete plays, timeliness of play, and safety.

Keywords: Computer games, Robotics, Calibration, Vision system, Manipulator

1 Introduction

Computer games are nothing new—they have been around as long as people have had access to computers. However, computer games are usually played in an artificial environment created within the computer. In this situation, the computer mediates all of the interactions between the players (one of which may be the computer. It is the computer that dictates the form of the interaction. Limitations of human-computer interaction constrain the user to mouse clicks, key presses, joystick movement, and in some rare cases, limited speech input. This is true even if the computer game provides a reasonable visual simulation of the real game. Many of these constraints are removed when the game is played over the table, with real rather than simulated playing pieces. The limitations become governed by what the computer is physically capable of doing in terms of the real world sensing and manipulation of the playing pieces.

The goal of this project is to enable a computer to autonomously play a board game over the table against a human opponent in much the same way as two people would play. The game is moved out of the simulated environment within the computer, and into the real world—our natural environment. For autonomous game playing, all of the interaction between the robot (the computer player) and the human player must take place through the medium of the game. Once the system is set up it should operate naturally, without user intervention. The only inputs from the human player are those normally required to play the game, by moving the playing pieces or whatever mechanism used by the game to play.

This paper will restrict the discussion to two player board games, although this may be generalised without too much difficulty to multiplayer games, and other table games. The success of the project can be judged by the degree to which it enables a computer to play seamlessly against a human opponent in the real world. The system could never be an unqualified success because of the constraints and limitations that must be imposed on the dialog to facilitate the interaction between the human and computer. In this regard, the computer system may be thought of as a physically and visually handicapped player. The success of the interaction may therefore be judged by the extent to which the interaction is sufficiently natural and intuitive for the human player to adapt to.

1.1 Phases of Game Play

A two-player board game can be thought of as a constrained dialog between the two participants. It is a dialog because the players are communicating with each other via the moves made, and the constraints are given by the rules of the game being played. The play of most games may be broken into three phases.

The first phase is negotiation, where players agree on their respective roles. For many games, this reduces to who is playing which pieces (for example black or white in Chess). An autonomous system must be able to participate in the negotiation phase, even if it is just passively. Take Chess as an example again. One approach would be for the robot to determine which colour it was from the orientation of the board, as convention in Chess dictates that players start with their pieces closest to them.

After negotiation, the next phase is the actual game play itself. Play generally alternates between the players, although in some games each turn may have multiple parts, for example making multiple jumps in Checkers. The moves within the game can be thought of as a conversation between the players, held in the language of the game. Information is exchanged between the players as each move is played. The game rules provide the context sensitive grammar that constrains the dialog between the players. In an autonomous system, all of the necessary information to play the game must be derived from the rules and the play of pieces.

The final phase involves recognising when the game has been completed. In general, it is clear when the game is over, for example in Backgammon, all of one player's pieces have been removed from the board. An autonomous system must be able to recognise when the game is over, including allowing the human player to resign gracefully. The system should make a smooth transition back to the negotiation phase if multiple games are being played.

The remainder of the paper outlines the requirements or key components of an autonomous game playing robot. For each component the particular implementation challenges are discussed.

2 System Components

There are at least four components or subsystems that make up a game playing robot. The first, and probably the most obvious, is to have a suitable game to play. The computer also needs an input system to enable it to sense the game state, and an output system so that it may physically manipulate the game state. Finally, the computer must have a module for controlling the dialog between itself and its human opponent. The controller module will generally incorporate a game engine which provides the rules of the games and selects a suitable move from the many that may be available.

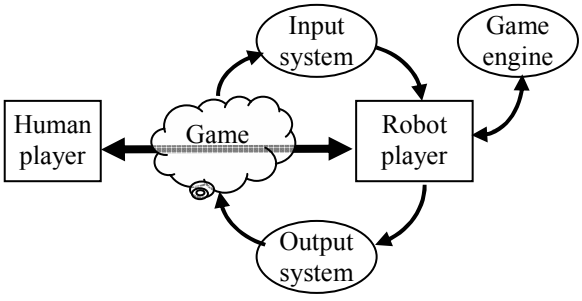

Figure 1: Interactions and subsystems required for a robot to play a game against a human opponent.

2.1 The Game: Trax

Trax was chosen for this project for a number of reasons. First, it is a two-dimensional game played with flat pieces. This significantly simplifies the design of the manipulator since it only has to move pieces in two dimensions. The manipulator has two principal axes of movement, with movement in the third dimension limited to enable playing pieces to be lifted over other pieces. Second, Trax is a relatively easy game to learn to play. There are only a few rules, and they are straightforward. Most people can learn to play Trax in about five minutes. The third reason is that in spite of the simplicity of its rules, Trax is a game of considerable strategic depth [1]. While Trax is easy to learn, it is by no means a trivial game.

Trax is a two-player abstract strategy game, invented in 1980 in New Zealand by David Smith [1,2]. It is played on a flat surface with square tiles with curved sections of black and white lines on one side, and straight lines on the other side (figure 2). One player plays black, and the other white, trying to form in their colour either a closed loop or line spanning from one side of the playing area to the other. Trax is a game of pure skill, since all of the pieces are identical and there is no luck involved.

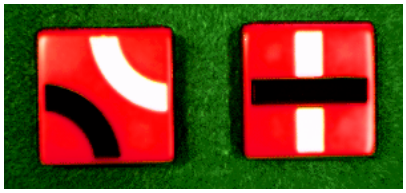

Figure 2: Trax tiles

In a turn, a player selects and plays a primary tile. This is played either side up against the tiles already in play, in such a way that the colours match where the paths join. As a result of playing the primary tile, there may be forced plays. Whenever two paths of the same colour enter a vacant space, a forced tile must be played that links the paths. Forced plays may also result in additional forced plays. Any such forced tile is played as part of the same turn as the primary tile. Therefore a turn may consist of several tiles being played. In fact it is these forced plays that give Trax its considerable strategic depth.

One constraint that was imposed on the robot was that it play with standard, unmodified Trax tiles. This constrains both the input and output systems, as discussed in the next section.

2.2 Game Engine

For the robot to play Trax, it must have a game engine. This analyses the current state, or game position, and determines what move to play next. The best move includes the primary tile, and any associated forced tiles. The game engine therefore incorporates the intelligence normally associated with conventional computer game.

Traditional game playing techniques, such as min-max or alpha-beta searching [3] do not work very well with Trax. A simple evaluation function using lengths and orientations of each path is ineffective at

recognising some of the complex threats used by intermediate to strong players. Such threats are typically characterised by sequences of up to 20 moves, requiring considerable search depth. A simple evaluation function cannot recognise such winning sequences until close to the ultimate move. Consequently, if the search is truncated before recognising the win, the evaluation can be inaccurate, and moves that initially appear reasonable may actually be faulty. Move selection and quality of play are therefore very sensitive to the number of search plies, or levels of lookahead. This problem is commonly referred to as the horizon effect.

This problem is exacerbated by the wide branching factor. In the middle of the game, each ply (or turn) may require choosing between 60 to 80 different moves. The high branching factor severely limits the search depth, making the horizon effect even more of a problem.

To overcome this, it is necessary to use a very complex evaluation function. As most of the threats in Trax are built from patterns of tiles [2], a form of syntactic pattern matching is used to recognise advanced threats in a single ply or level of lookahead. The internal representation of the game state is therefore optimised for this pattern matching.

Pattern matching is implemented using an augmented finite state machine. The state machine approach allows related sub-patterns to be grouped together and recognised efficiently. When a threat is recognised, the state machine provides information on how many turns the threat takes to win, and what the next move in the sequence is. Lookahead is then used to verify that the complete threat works. At each ply, the move is made, and the position re-evaluated. Each time the number of moves required should decrease. If not, either the threat is faulty (one of the moves cannot be made because it results in a win for the other player) or the particular variation does not work. The playing level of the game playing system can be adjusted by ignoring patterns above a certain level of complexity.

While lookahead is ineffective for threat recognition in Trax, it is useful for more general strategic play. At the higher playing levels, two or three ply of general lookahead are combined with pattern matching. In both cases of lookahead (general and for threat verification) the 'killer heuristic' [3] is very effective at pruning the search tree.

2.3 Input System

To interact with the human opponent using real tiles, the robot must have some means of sensing the current game state. For Trax this requires detecting the locations of the tiles within the playing area, and hence the current state of the game. It is also necessary to detect the locations of the unplayed tiles around the playing area, as these constitute the pool of tiles available for making the moves.

To make the playing experience as natural as possible for the human player, it is desired to place as few constraints as possible on the positioning and orientation of the tiles. Although the tiles in the playing area form a regular grid, the absolute position and orientation of the grid is unimportant to the game, so should be left unconstrained.

Using standard, unmodified Trax tiles requires some form of vision system for sensing the location and orientation of the tiles. A single, global, camera constrains the interaction area to the field of view of the camera. This must be larger than the extent of manipulation of the output system, to enable the input system to detect any changes that the output system is physically capable of making. The resolution of the camera will limit the accuracy with which the computer is able to locate the tiles.

The ideal position for the camera would be directly above the playing area. Unfortunately the smooth surface of the tiles makes them prone to specular reflections. To prevent the tiles from being washed out, the lights would need to be placed down low, and become obtrusive.

The alternative is to view the playing area from one side and light the playing area from the same side (figure 3). In this way all of the specular reflection is directed away from the camera.

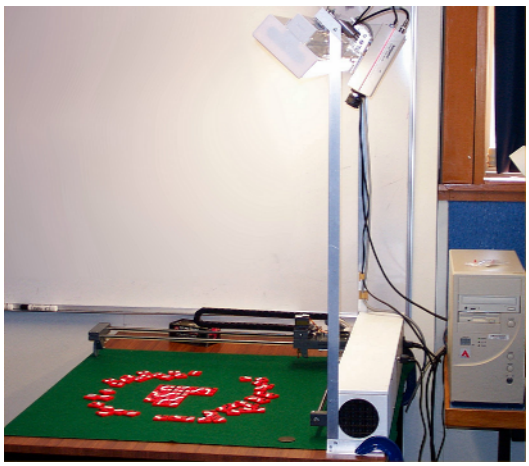

Figure 3: Side view of robot, showing the position of the lights and camera.

A consequence is that the image is now subject to significant perspective distortion. The short focal length lens also introduces significant barrel distortion. Both of these effects are clearly seen in the image captured by the camera (figure 4).

Both distortions may be characterised and corrected using a method derived from Bailey [4]. While it is possible to detect the tiles in the distorted image, and transform the parameters, the variation in scale caused by the distortions makes it simpler to transform the complete image prior to detecting the tiles. Once transformed, the tiles are all uniform size, as shown in

figure 5. The resolution in the transformed image is approximately 1.2 mm per pixel.

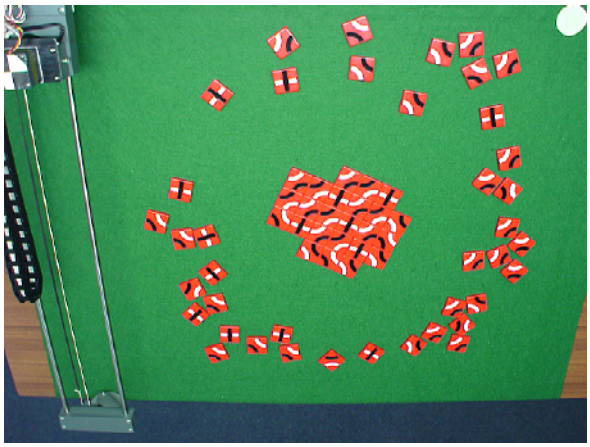

Figure 4: The camera's view of the playing area. Note significant perspective and barrel distortion.

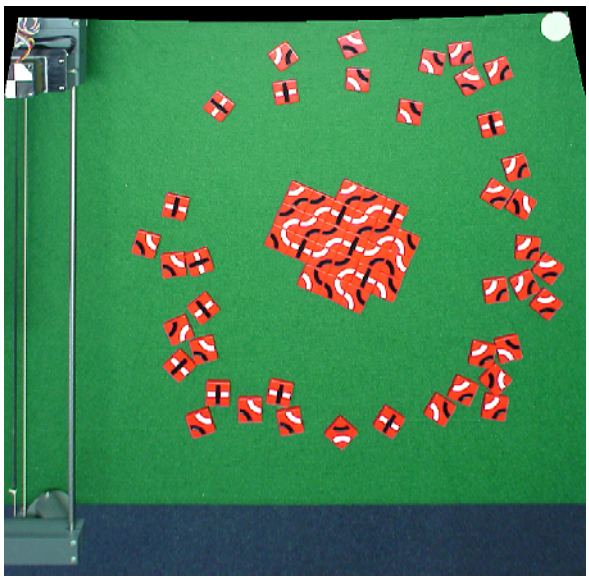

Figure 5: The image from figure 4 after correcting for perspective and lens distortion.

Although the tiles have high contrast, the resolution is inadequate to reliably detect the edges of the tiles, especially when the tiles are placed adjacent to one another. The oblique camera angle and 6 mm tile thickness also means that the side of the tile is also visible, particularly for the tiles that are furthest from the camera. Reliable detection of the location and orientation of the tiles is therefore based on detecting the patterns of black and white paths on the tiles.

The segmentation process is described more fully by Bailey [5], but is listed briefly here for completeness. The first step in this process is a segmentation into tile, table, black and white path regions by examining the red and green components of the colour image. Template matching, using a simplified 4 point template is used to locate candidate tile positions. Signatures around circles of radius 5 and 9 pixels are used to reject false tiles. Then using all black and white pixels within 12.5 pixels of the candidate centre, the true centre and orientation of the tile are determined to subpixel accuracy using first and second moments. The output from the image processing module is a list of centres and orientations of each tile found.

Another task of the vision system is to determine when the playing area is occupied by the human player. The difference between successive images (about one second apart) is used to determine when the playing area is clear. If the number of pixels that are different is below a threshold, the playing area is assumed to be free from obstruction, and the position analysed to determine who it to play next.

2.4 Output System

The computer must also have some means of physically manipulating the game state. When it is the robot's turn to play, it needs to be able to pick up the tiles from the pool of unplayed pieces around the periphery of the playing area, and place them adjacent to the existing tiles in the correct orientation.

Again, because standard tiles are being used, this implies some form of arm and gripper to pick and place the tiles. Although the playing area is 2 dimensional, it may be necessary to lift tiles over other tiles to get them into place. This and the fact that tiles usually need to be rotated to get them into the correct orientation require a minimum of 4 degrees of freedom.

The biggest constraint imposed by the manipulator system is the limited range of motion that is possible if the base of the arm is fixed. While Trax potentially may be played to an unlimited size, practical considerations impose a limit on the number of tiles that may be played, and the size of the playing area. Trax tiles are 32 mm square, and most games are completed within a 10x10 tile playing area. However the first tile played may actually be on the edge of the playing area. Therefore to enable most games to be completed, a practical playing area must be at least 0.5 m square, and preferably larger.

These constraints are met by using a two-axis (X,Y) gantry robot for the primary movement. Additional axes are provided for Z (to lift tiles) and theta (for rotating the tiles). All four axes are controlled using stepper motors, with precise positioning maintained in the X and Y directions by toothed belts. The use of stepper motors enables open loop control of the motors. Table 1 lists the range and resolution of each of the robot axes, and figure 6 illustrates the relative positioning of the motors in the assembly.

Table 1: Range and resolution of each axis.

Axis	Range		Resolution
X	622 mm	5880 steps	0.11 mm
Y	542 mm	2565 steps	0.21 mm
Z	24 mm	180 steps	0.13 mm
Theta	338 degrees	375 steps	0.9 degrees

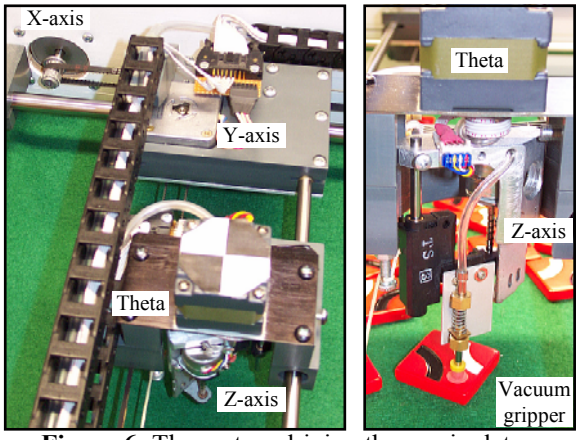

Figure 6: The motors driving the manipulator. The right panel shows the details of the head.

The stepper motors are all controlled by a single 8051 microcontroller. Communication between the host computer and the 8051 motor controller is via a 19200 baud RS232 serial connection. The host PC determines what actions are required in terms of the number of steps for each stepper motor. It then sends movement instructions to the 8051 via the serial connection. The limited resources on the 8051 mean that commands cannot be queued. The 8051 therefore acknowledges receipt of a command when it is sent, and sends a command complete message when the movement is complete.

Several alternatives were considered for the gripper. Finger-like grippers were eliminated because of the necessity to place the tiles directly adjacent to other tiles. This leaves little or no space for fingers beside the tiles. Standard Trax tiles are made of plastic, so a magnetic gripper is unsuitable. The gripping problem is complicated further by the fact that the paths on the tiles are recessed below the tile surface, complicating the use of a vacuum gripper. Through experimentation, it was found that 6 mm diameter surface mount vacuum pick-up cup was able to hold the tile from one corner (figure 6). The vacuum is controlled by switching a small diaphragm pump on and off. The vacuum cup is spring loaded with about 10 mm of travel. This provides compliance for when the table may be slightly warped, and prevents damage to the gripper if the positioning is not accurate.

One aspect of playing Trax that the current manipulator cannot handle is the flipping of tiles. Because the tiles can be played either side up, it may be necessary to flip a tile as it is played. In most Trax games, approximately 80% of the tiles played are curved side upwards and 20% are straight side upwards. By starting with tiles in these proportions, most games will be able to be played.

An important aspect of any human-robot interface is safety. This is especially important in a game where both the human and the robot share the same workspace. The use of low powered stepper motors means than in the event of a collision, the stepper motors will skip steps rather than continuing on. Without shaft encoding on the motors, the only way of recognising collisions is through feedback from the vision system. To this end the input system can periodically check that the manipulator is in the expected position.

2.5 Calibration

The manipulator, vision system, and game engine all have their own coordinate systems: the manipulator is in steps, the vision system in pixels, and the game engine uses logical tile coordinates. It was decided to use a common coordinate system aligned with the manipulator, but measured in mm rather than steps.

Calibration is required to determine the correspondence between points seen by the input system, and points on the table. The simplest way of achieving this would be to have a grid positioned on the table; however this would require precise alignment when setting the system up. The alternative chosen was to move the manipulator to known positions within the field of view, and create a map that way.

A simple target has been mounted on the back of the manipulator directly over the centre of rotation, as seen in figure 6. The target allows subpixel location of the manipulator in the field of view using a simple matched filter. The manipulator is moved in a 3x3 grid pattern and images captured at each of the vertex locations.

A parabola is fitted to each set of 3 co-linear points. Since perspective distortion transforms straight lines to straight lines, the quadratic term provides information on the lens distortion [4]. Once this is characterised, the perspective transformation is determined to map the lines to their known locations.

This gives the calibration to the plane at the top of the manipulator, not the plane of the tiles, which is about 20% further from the camera. The scale factor and offset to the tile level is determined by locating a tile at the zero location, and using the manipulator to move it to the opposite corner (a known distance).

When a game is started, the tile in the centre of the field of view is found. Adjacent tiles (within 5 mm) at a similar angle are added. After all such tiles are added, the affine mapping between system units and logical tile units is determined by least squares fit. This allows the game to be played at any orientation and position, as long as it is in the centre of the field of view.

2.6 Coordination

The final module is the coordination module which controls the specific activity of the robot based on the current context within the game being played. Making

the playing of the game natural and intuitive for the human player is perhaps the most challenging aspect of this project. To reduce the possibilities of misunderstanding between the robot and the human player, the robot is generally passive, responding to actions made by the human.

When two humans play Trax over the table, they typically start a new game by clearing the centre of the table of tiles. This same action was used with the robot. The human player starts a new game clearing the centre of the table of tiles, beginning the negotiation phase. For the human's benefit, the centre of the table is marked with a small label.

In the negotiation phase, the players determine who will be represented by the black paths and who will be represented by the white. This negotiation is achieved by the human player informing the robot which colour it is playing by placing a small disk in the top right area of the field of view. When a new game is started, the robot looks for this disk. If absent, or if the disk is black, the robot plays black, otherwise it plays white. This is only sampled when a new game is detected (that is during the negotiation phase).

Audio feedback is provided to the human player whenever a new game is started, and whenever the robot's colour is changed. Although the robot is passive during the negotiation, this acknowledgement was found to be important to make the playing experience more natural.

If the robot is playing white, it automatically plays the first tile in the centre of the playing area. When the computer has finished its move, it retracts the manipulator to the home location at the far edge of the playing area. It then waits for the human player to complete their turn and move their arm from the playing area. The forced play rule takes beginners a few games to learn, so if not all forced tiles have been played, the robot reminds the human to complete their turn.

At present, the robot will only make a move if the position is valid (all forced tiles have been played) and there are more tiles present than at the end of its previous turn. This, however, allows the human player to cheat by rearranging tiles. With conventional computer games, this is not a problem because the computer is the arbiter of what is valid. Indeed, the computer is usually programmed to prevent a human opponent from making an invalid move. However, a robot playing over the table has no such control. It is much harder to form an appropriate response to invalid human play.

While deliberate disruptions must be recognised, they are considered to be of minor importance because any game play requires the cooperation of, and collaboration between, the players in addition to the competitive aspect. The simplest response to a disruptive player is to refuse to play until they make a valid move. One refinement to the robot would be to check the position to determine if it can be achieved in one turn from the previous position. This will ensure that the human player also follows the rules.

At present no check is made that all of the tiles that were meant to have been played by the robot actually were played (occasionally a tile is not picked up cleanly or is dropped). It would be a straight forward matter to check the integrity of the game after play, and retry the tiles that were missed. The controller should also check that the robot is indeed back in the home position after the move in case the arm was bumped and steps were skipped. This may be achieved by detecting the target and checking that it is indeed in the home position.

The third and final phase is the end of the game. This is normally when one or other player has achieved their objective. However, it is also necessary for the human to be able to resign gracefully without actually completing the game. This is accomplished by allowing the human to reset the game by clearing the centre of the playing area.

3 Conclusions

The current Trax playing robot extends computer games from out of the virtual domain of the computer, and onto the table, into our real world domain. The user interface to the game is no longer through mouse, key strokes and a video screen, but through the medium of the physical game pieces. As a result, the playing experience is quite natural—almost like playing against another person over the table.

Perhaps the most significant difference between playing against the robot and against another human opponent are the constraints to the playing area. The restricted range of motion requires the game to be played near the centre of the field of view.

At present, the robot is a little clumsy, frequently placing tiles slightly over the top of other tiles. Part of the cause of this relates to the limitations of the calibration process. In spite of this, it is quite natural to play Trax against the robot over the table.

4 References

[1] D.G. Bailey, *Trax Strategy for Beginners*, 2nd edition, Donald Bailey, Palmerston North, 1997.

[2] D. Smith, *How to Play Better Trax*, David Smith, Christchurch, 1983.

[3] D. Levy, *Computer Gamesmanship*, Century Publishing, London, 1983.

[4] D.G. Bailey, "A new approach to lens distortion correction", *Proceedings Image and Vision Computing New Zealand*, pp 59-64, 2002.

[5] D.G. Bailey, "Vision System for a Trax Robot", *Proceedings of Image and Vision Computing New Zealand*, pp 354-359, 2003.