Trax Playing Robot

Donald Bailey, Ken Mercer, Colin Plaw, Karthik Subramaniam
Institute of Information Sciences and Technology
Massey University
Palmerston North
New Zealand
Email: D.G.Bailey@massey.ac.nz
http://www-ist.massey.ac.nz/DBailey

Abstract—A Trax playing robot is described that is designed to play the two-player strategy game Trax against a human opponent using physical tiles. The robot consists of four main modules. The vision system uses a USB web-camera to capture images of the playing area and processes the image to extract the current state of the game. The playing system analyses the current game state, and determines the move to play. The manipulator system plays the move by physically placing the necessary tiles. This whole process is controlled by the coordinator system.

Index Terms—Trax, computer games, human vs machine, image analysis

I. Introduction

Computer games are nothing new—they have been around as long as people have had access to computers. What is novel with this project, however, is that we are building not just another computer game, but a robot that physically plays the game in much the same way that a human player would. Indeed, the goal of this project is to develop a system that seamlessly plays Trax against a human opponent.

A. Why 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 mostly 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. While Trax is easy to learn, it takes some time to learn to play it well.

B. What is Trax?

Trax is a two-player abstract strategy game, invented in 1980 in New Zealand by David Smith [1-3]. 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 (see figure 1). 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.

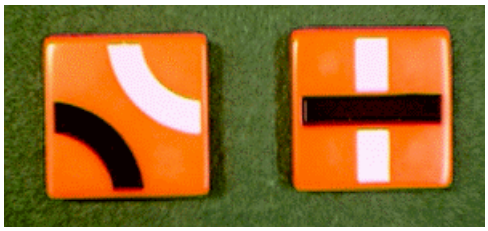

Figure 1: Trax tiles

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 tile is played as part of the same turn. Therefore a turn can consist of several tiles being played. In fact it is forced plays that give Trax considerable its strategic depth.

II. A Trax Robot

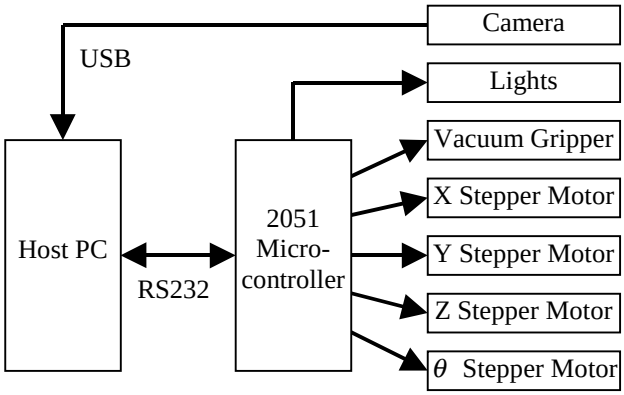

Figure 2: Trax robot hardware.

The hardware for the Trax playing robot is shown in figure 2. The main software is written in C++ on a host PC. Images of the playing area are captured through a web-camera via a USB port. The robot manipulator is

constructed in a separate box, and controlled using a 2051 microcontroller. This communicates with the host PC via an RS232 serial connection. Specific hardware details are described more fully in later sections.

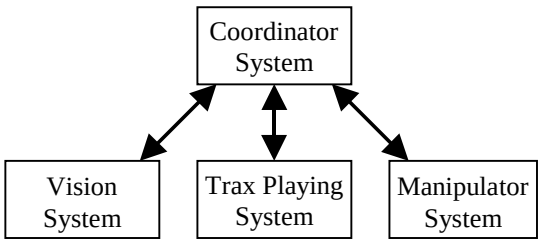

Figure 3: Software modules for the Trax robot.

The software for the Trax robot is divided into four modules as shown in figure 3. The vision system is responsible for capturing views of the playing area and surrounding tiles, and determining the location of each tile. Those tiles that are currently in play are represented in a form that enables the Trax playing system to determine the next move to play. Once a move has been determined, the manipulator system is instructed to move appropriate tiles from the surrounding unplayed area into the playing area adjacent to the existing tiles. This whole process is under the control of the coordinator system.

Each of these modules will be described in more detail in turn.

A. Vision System

The vision system captures and processes images of the playing area. It uses a camera connected to the host PC via a USB port. Single images are captured on demand through a TWAIN software interface.

The primary task of the vision system is to determine the location and orientation of all visible tiles. Typically a game is played in the middle of the table, and unused tiles are scattered around the edges (see for example figure 4). It is necessary to determine which tiles are in play and which tiles are available for play.

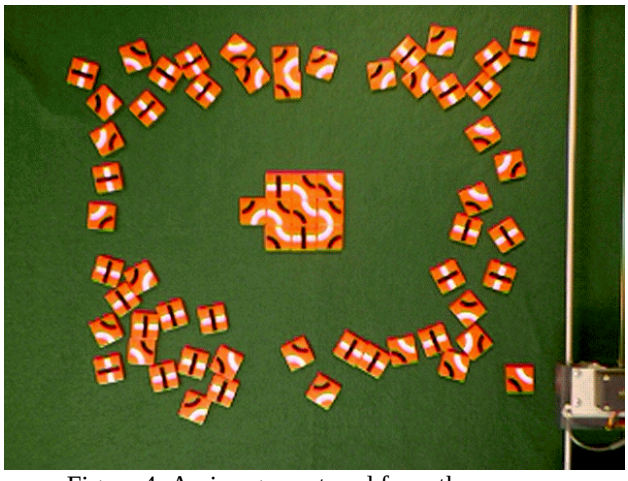

Figure 4: An image captured from the camera.

The field of view of the camera is approximately 750 mm by 550 mm, which is captured within an image size of 640x480. As a result, the camera resolution is about 1.2 mm per pixel. Although the tiles have high contrast, this resolution is inadequate to reliably detect the edges of the tiles when the tiles are placed adjacent to one another. Since the camera uses an inexpensive, single CCD sensor, the resolution in each of the colour planes is significantly reduced. This is most noticeable in the red and blue planes which appear significantly blurred compared with the green plane.

While the red plane provides a strong indication of the location of the tiles, even when the tiles are isolated the detected edges can be 2 to 3 mm out. The errors result from at least two sources. The blur in the red plane makes the uncertainty in the location of the edges to ±2 pixels. The camera has a wide angle lens, and is positioned approximately 900 mm above the table. Therefore the tiles at the edge of the table are not viewed vertically, but at an angle of up to 20 degrees. The 6 mm tile thickness results in tile appearing up to 2 mm wider in the radial direction. Therefore detecting the red associated with the tiles only provides an indication of the location of the tile and not an accurate measurement.

Since the background of the playing area is green, an inexpensive computation for tile segmentation is therefore to compare the red and green planes. Those pixels with a greater red pixel value are classified as tile, and those with a greater green value are classified as background.

However to determine the precise location and orientation of the tile, it is necessary to locate the black and white segments on each tile. For this, a weighted sum of the red, green and blue planes provides the best resolution. The local average within a 25x25 window (about the size of each tile) is subtracted from each pixel, and the difference thresholded. Those pixels with differences less than –64 are classified as black paths, and those with differences greater than +64 are classified as white paths (these threshold levels have to be adjusted depending on the light level). The results of this processing applied to the image in figure 4 are shown in figure 5.

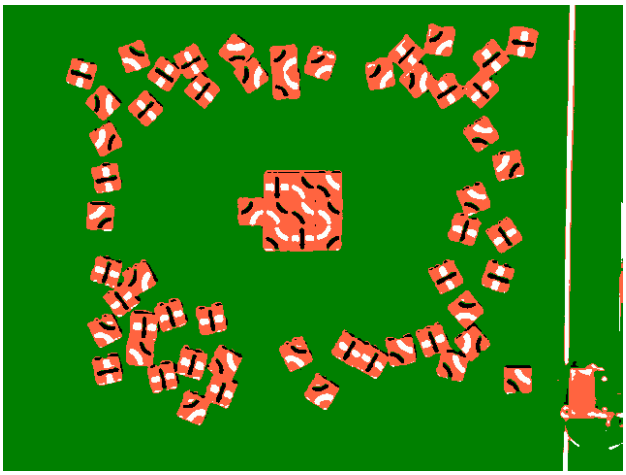

Figure 5: Captured image after simple segmentation.

The few black pixels around the edge of the tiles are a result of shadows, and the white pixels on the edge of the tiles result from specular reflections. Both shadowing and specular reflections are a problem and are very difficult to

eliminate completely. Diffuse lighting minimises the shadows, but because both the tiles and the paths on the tiles have a strong specular component to their reflectivity diffuse lighting from close to the camera is unsuitable. The reflectivity problem is made worse by the fact that the camera has a wide angle of view, requiring the lights to be a significant distance from the camera.

An alternative approach that is being considered is to have the camera to one side, and lighting from behind and below the camera. This effectively moves the zone of strong specular reflection out of the field of view. The disadvantage however, is that it introduces significant perspective distortion. The camera resolution is already marginal when looking vertically. With the perspective distortion, the pixel size on the side of the playing area away from the camera increases to about 1.5 mm.

After the initial segmentation, the next processing step is to pair each black segment with its corresponding white segment. The shape of the segment indicates whether the tile has the curved or straight side upwards. Finding the centre of gravities of each of the paths allows the tile location to be determined to sub-pixel accuracy [5].

A secondary task of the vision system is to determine when the field of view is obstructed. This will occur when the human player is making their moves. Images are taken 5 to 7 seconds apart and the difference between the images is calculated. Significant differences between successive images imply that a human player is moving within the field of view. While this is a relatively crude indication of whether or not the playing area is clear, it is considered adequate for most circumstances.

B. Trax Playing System

The game playing system determines the best move to play, given a current position. The best move includes the primary tile, and any associated forced tiles.

Traditional game playing techniques, such as min-max or alpha-beta searching [6] do not work very well with Trax. While a simple evaluation function that uses the lengths and orientations of each path is very effective against beginner players, they are completely ineffective at recognising some of the complex threats used by more advanced players. These threats are typically characterised by sequences of up to 20 moves requiring considerable search depth. When a simple evaluation function is used, the Trax playing system cannot recognise such winning sequences until close to the ultimate move. As a result, such an approach is very sensitive to the horizon effect. This is where the search is stopped at a certain point, and the evaluation of the position is inaccurate as a result of the truncation. Moves that appear reasonable initially may actually be faulty, but the search stopped short of actually determining this. As a result, move selection and quality of play are very sensitive to the number of search plies.

This problem is exacerbated by the wide branching factor. In the middle of the game, each ply can require choosing between 60 to 80 different moves. For example, the position shown in figures 4 and 5 (which is near the start of the game) has 30 different moves that can be played. 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 recognise advanced threats in a single ply. The internal representation of the game state is optimised for this pattern matching.

Pattern matching is implemented using an augmented finite state machine. The state machine approach allows related patterns to be grouped together and recognised efficiently. When a threat is recognised, the state machine provides information on how many plies that the threat takes to win, and what the next move in the sequence is. Lookahead is then used for threat verification. 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’ [6] is very effective at pruning the search tree.

C. Manipulator System

The task of the robot manipulator is to pick up tiles that are not in play and place them in the game playing area when it is the robot’s turn to play. The two-dimensional nature of Trax makes a two-axis (X,Y) gantry robot suitable for the primary movement, as illustrated in figure 6.

One advantage of an (X,Y) manipulator is that the coordinate system is rectangular, considerably simplifying the calculations required to move the manipulator to a particular point. It also allows a much greater reach from a small, compact base than a more anthropomorphic ‘arm’. The gantry system can easily cover the 700 mm x 500 mm playing area with a total height of less than 150 mm. The biggest disadvantage of a gantry system is that the ‘arm’ cannot retract completely from the active area. It is always present on the edge, even when in its retracted ‘home’ position.

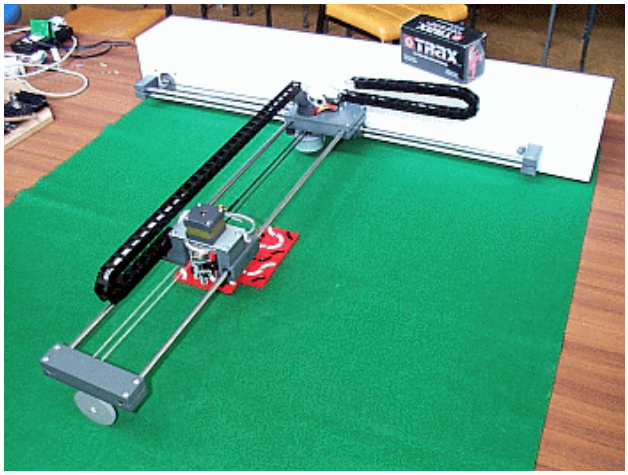

Figure 6: Side view of Trax robot manipulator.

Mechanical stability is achieved through the use of linear bearings. The X-axis uses a single 12 mm diameter shaft with a pair of bearings separated by 100 mm. The gantry is moved in the X direction from one end, with a wheel providing support at the other end. The Y-axis, along the gantry, uses two 8 mm diameter shafts separated by 100 mm. The manipulator 'head' is mounted on the two shafts with a single linear bearing on each. The two separated shafts provide reasonable stiffness.

Stepper motors control movement of the gantry along the X-axis and the head along the Y-axis. Each of these motors has a resolution of approximately 0.5 mm per step. Precise positioning is maintained through the use of toothed belts to drive the moving components from the motors.

Tiles also have to be picked up, rotated and placed down again. The manipulator 'head' provides these functions. 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 or other similar 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 (see figure 7). Vacuum is controlled by switching a small diaphragm pump on and off.

Two further stepper motors provide the lift and rotation. The Z-axis motor, shown in figure 7, provides up to 20 mm movement – more than enough for lifting a tile over other tiles that may be in the way. The resolution of the Z-axis is not important, and the motor only needs to provide sufficient power to lift a tile.

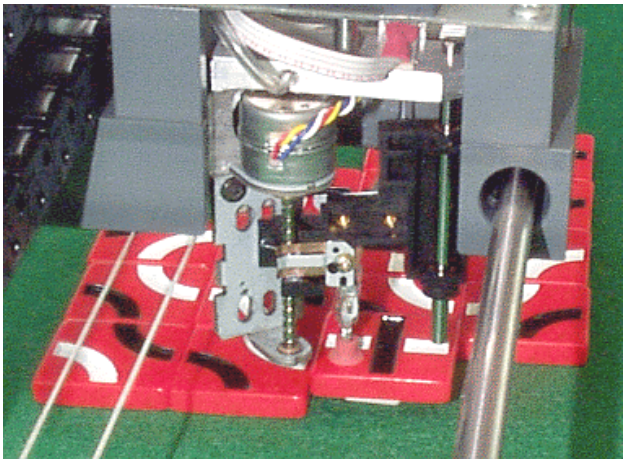

Figure 7: Detail of robot manipulator 'head'.

Tile rotation is accomplished rotating both the Z-axis motor and the gripper using a 4th stepper motor. A 200-step motor provides an angular resolution of 0.9 degrees when operated in half-step mode. This resolution corresponds to an offset of ± 0.5 mm over the length of the tile, and is consistent with the resolution of the X- and Y-axes. Connections for the Z-axis motor and vacuum to the gripper constrain the rotation to about 270 degrees. This is more than adequate because it allows the tile to be rotated through any arbitrary angle by rotating the gripper prior to picking up the tile. To simplify computation of tile positioning, the centre of rotation corresponds to the centre of the tile. This means that the gripper is offset 13 mm from the centre of rotation.

The robot manipulator is controlled by a 2051 microcontroller. The host PC determines what actions are required in terms of the number of steps required for each stepper motor. It then sends movement instructions to the 2051 via an RS232 serial connection. Moving a tile involves the following sequence of actions:

1. X, Y, θ —The head is moved over the tile that is to be picked up. It is rotated to correspond to the orientation of the tile
2. Z—The gripper is lowered until it contacts the tile
3. V—Vacuum is applied to grip the tile
4. Z—The gripper is raised
5. X, Y, θ —The head is moved to the location and orientation where the tile is to be played.
6. Z—The gripper is lowered
7. V—Vacuum is turned off, releasing the tile
8. Z—The gripper is raised
9. X, Y, θ —If there are further tiles to play, the head is moved to the location and orientation of the next tile, otherwise the head is moved to the home location to allow the human player to make their move.

One aspect of playing Trax that the current robot cannot handle is flipping of tiles. Because the tiles can be played either side up, it may be necessary to turn 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.

ISBN 962-442-191-9 © M2VIP 2001

D. Coordinator System

The role of the coordinator system is to ensure smooth interaction and communication between the other modules.

Each of the other systems has its own coordinate system, so the first task on initialisation is to calibrate each module to determine the relationships between each set of coordinates. Since the tiles are physically moved by the manipulator, and the manipulator uses a rectangular coordinate system, it makes sense to use this as the base.

The vision system makes all measurements in terms of pixel units. Having the camera at a fixed height enables calibration to be performed directly by moving the gantry within the field of view. This enables calibration of both their relative position and perspective distortion resulting from camera angle. To do a full calibration of lens distortion requires capturing 20 to 30 images with the head in different positions. Calibration is simplified by the fact that all motion is in a plane.

The internal representation of the game state requires only the relative positions of the tiles in terms of virtual tile coordinates. The position and orientation of the first tile played determines the required transformation, although this can be refined as further tiles are played. Transformation between virtual tile coordinates and manipulator coordinates involves rotation and scaling.

Once calibrated, the primary role of the coordinator system is to determine whose turn it is (whether robot or human) by analysing the tile data from the vision system. This requires periodically requesting data from the vision system. An assumption is made that the game is played in the centre of the field of view. Therefore the largest group of tiles in the centre of the image is located and these tiles defined as 'in play'. The remaining tiles around the periphery are currently unplayed, and are available for either the human or the robot to play in their turn. The coordinator system constructs a representation of the game state from the tiles in play.

If there are obstructions in the field of view, it is assumed that the human is playing. Once the field of view is clear, the game position is analysed to determine if the human opponent has completed their turn.

The robot does not make its move until all of the following conditions are met:

1. the current playing position is complete (all necessary forced tiles have been played),
2. it is the robot's turn to play, and
3. the field of view is not obstructed by the player.

When it is the robot's turn to play, it then passes the current game state to the game playing system, which returns the move to play. The nearest tiles to the final location with the right side upward are then selected for play. The manipulator then moves them into position, and the cycle is repeated.

III. Discussion

Perhaps the most complex task of a robot game playing system is to recognise and respond appropriately to invalid human play. In Trax, this requires recognising unexpected tile movements, either because the tiles within the playing area were bumped, or tiles were deliberately removed from or added to the game. Bumped tiles are reasonably easily recognised, and the system can readily keep track of moves as they are played enabling minor errors in reconstructing the position to be recognised. If necessary, the expected position can be displayed on the monitor of the host PC.

While deliberate disruptions must be recognised, they are considered to be of minor importance because any game play requires the cooperation and collaboration between the players. The simplest response to a disruptive player is to refuse to play until they make a valid move.

One added complication with Trax is that a move can involve playing more than one tile. It usually takes several games before new players learn to recognise and play all of the required forced tiles. Some form of feedback will be necessary to indicate to a human player that their turn is not complete. It is important that each player play their own forced tiles, otherwise new players can get confused by thinking their opponent is having several turns.

An important aspect of any human-robot interface is safety. This is especially important in game playing where both the human and the robot share the same workspace. The use of low powered stepper motors mean than in the event of a collision, damage is minimised. If the load is too great on the motors, they 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. Any collision will require the manipulator to move back to its home position to recalibrate.

The Trax playing robot demonstrates the coordination between a vision system, an artificial intelligence planning module, and a robotic manipulator. Its purposes are two-fold – to promote interest in the game, Trax, and to promote interest in students in mechatronics and image processing.

IV. References

[1] D. Smith, How to Play Better Trax, David Smith, Christchurch, 1983.
[2] D.G. Bailey, Trax Strategy for Beginners, 2nd edition, Donald Bailey, Palmerston North, 1997.
[3] http://www.traxgame.com
[4] TWAIN Specification, version 1.9, TWAIN working group, January 20, 2000.
[5] K. Subramaniam, Trax Game-Playing Vision System, BEng degree project report (D.G. Bailey supervisor), College of Sciences, Massey University, 2000.
[6] D. Levy, Computer Gamesmanship, Century Publishing, London, 1983.