Making Real Games Virtual: Tracking Board Game Pieces

Steven Scher, Ryan Crabb, and James Davis
University of California at Santa Cruz
{sscher, rcrabb, davis}@ucsc.edu

Abstract

The same game is often played in real and virtual worlds. We integrate in-person and on-line playing of board games such as Go, bringing the real world into the virtual world. A player may record an in-person game by placing their camera on the table next to the game board, taking photos of the game. After automatically detecting the board and playing pieces, we perform inference on the time series of detections to eliminate errors and accurately estimate long sequences of moves. The game transcript may be studied afterwards, shared with friends and teachers, or added to online compilations, bringing the attendant benefits of online game play to an in-person game.

1. Introduction

1.1. Motivation

Aficionados of many board games, such as the classics Chess and Go, often follow games by discussing particularly good and bad moves with each other, with friends, and with teachers. This commentary is a large part of the social scene and central to skill level improvement. These games are often played both in-person and online. While the events of virtualized games are easily transcribed, the real-world game does not so easily lend itself to review and archival.

Robustly transcribing the moves of a game from photos in a real world environment is made challenging by changing lighting, transient shadows, and occlusion by players' hands. Prior work has identified playing pieces in a single photo when none of these obstacles are present, but no method has yet addressed the transcription of an entire game in a natural environment.

We present a method to transcribe the moves of a board game in the real world, automatically and unobtrusively. A player simply places a camera on the table next to the board, recording a series of photos; our algorithm automatically finds the board and detects moves, creating a complete transcription of the game.

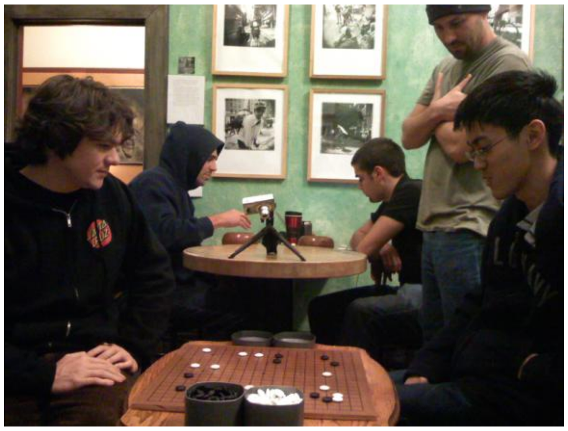
Figure 1. A game of Go is played in a cafe, recorded by a digital camera on an adjacent table.

1.2. Our approach

In games of perfect knowledge, the entire state of the game is known to all players. Board games represent this state visually on the board. Typically, with limited sets of distinct playing locations and types of pieces, a single complete picture of the game is sufficient to determine nearly the entire state of the game.

The playing board is automatically detected in the photos without user input, and the positions where game pieces may appear are found. The game pieces present are detected, and misclassifications are eliminated by finding the most likely sequence of legal moves.

Our contribution is a method to transcribe a real world board game from photos, combining computer vision techniques with inference to create a robust system. This approach is applicable to many games, as demonstrates by its application to the game of Go.

This paper is organized as follows. Section 2 summarizes related research. Section 3 describes the algorithm in detail, and section 4 presents empirical tests. Section 5 provides further discussion and directions for future work.

2. Related work

The task of recording Go games with a camera requires two major elements. First, the board must be automatically identified. Second, each possible location of a stone must be classified as a black stone, white stone, or an empty intersection.

The Go board is a 19x19 rectangular grid of black lines on a wooden board. Camera calibration techniques, such as [5], provide a framework for developing a board detection algorithm. The Hough accumulator representation of the detected lines may be directly used to detect the grid, as in [1] and [6]. In another method [1], line intersections are detected by classifying SIFT features, and another performs a genetic algorithm search [3].

Classifying image patches into a small set of possible objects is a well studied problem [8]. To detect Go stones in single photos, the neighborhood brightness is thresholded in [2] and [4], and a cascade of three classifiers, is applied in [1]. Reported results are insufficiently reliable for an automatic whole-game recording system.

3. Methods

3.1. Detecting the board

The board is found automatically by detecting edges, then lines, then a subgrid, and finally the full-size grid. First, edges of all orientations are found with a radially-symmetric laplacian filter. Lines found with a Hough transform line detector are clustered into two groups according to their orientations using K-means. For each cluster, edges at the dominant orientation of the cluster are found with an oriented 1D Laplacian filters, and lines are found with a Hough transform and maximum-likelihood refinement.

RANSAC is applied to find a confidently-identified subgrid of the whole grid. Two lines of each orientation form a rectangle; a rectangle and a guess at grid spacing define a homography. Each guessed homography is scored by how well the implied gridlines match detected edges.

The best-scoring subgrid is chosen and greedily grown by hypothesizing new grid lines in each direction and accepting the best match until reaching the full 19x19 size. Once the board is found in the first image, all images are warped to an overhead orthographic view and cropped. A median filter in the time dimension is applied to the image sequence to remove most of the players' hands.

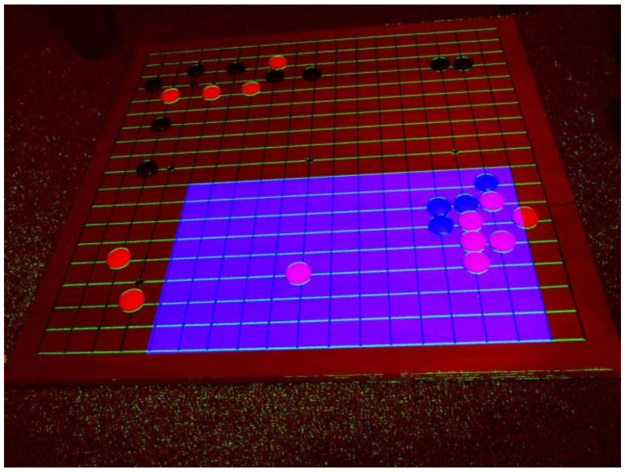
Figure 2. Lines are detected (horizontal lines shown in green) and a subgrid is found using RANSAC (blue). The subgrid is greedily grown to full size to detect the board.

3.2. Detecting stones

With the board detected and image rectified, the 19x19 grid of 361 locations where a stone might appear are known. A template of the expected stone shape and color is correlated with the image, and the maximum response to each template in the neighborhood of the candidate location provide classification probabilities.

This classifier, and any classifier operating independently on each time step, is prone to errors in the presence of shadows, occluding hands, and similar artifacts. Rather than adopting a more sophisticated classifier, we reduce errors by performing inference over time. Any imperfect classifier will benefit from the inference method described below.

3.3. Inference model and notation

In this section, we construct a hidden Markov model relating the true state of the board to the classification probabilities from 3.2. The HMM constrains the true sequence of states to be a legal sequence of moves according to the rules of the game.

For a general board game, consider variables L_1, L_2, …, L_N corresponding to each of the N locations on the board where a playing piece might be placed, and $\text{photo}_i(t)$ be the image patch of location i at time t. Each variable L takes on one of a small number of values to indicate what type of playing piece (if any) is present. Let the state of the entire board at time t be denoted B(t), an assignment of a value to each variable $L_i(t)$, i=1…N. Let a sequence S be an ordered set of T board states {B(1), B(2), …, B(T)}.

In the game of Go, we have N = 361 playing locations each with 3 possible values (black-, white-, or no-stone). The board state B specifies a value for each of the 361 locations, as well as extra variables to note the time and which player's turn is next. In transcribing a game, the sequence S corresponds to the state of the board in each picture 1…T. A sequence S ={B(1), B(2), …, B(T)} is a "legal sequence" if B(1) is the empty board, and B(t) is the result of a legal move from state B(t-1) for t=2…N.

3.4. Probabilistic formulation

We find a maximum a posteriori solution of the most likely sequence given the evidence (photos), maximizing p(S|photos), out of all legal sequences. With each photo independent, this is the product:

(1) $p(S|\text{photos}) = \Pi_{t=1\ldots T}\, p(B(t) \mid \text{photo}(t))$

We can use Bayes rule to instead maximize:

(2) $\Pi_{t=1\ldots T}\, p(\text{photo}(t) \mid B(t)) \times p(B(t))$

Here, the prior p(B(t)) encodes only the distinction between legal and illegal board state sequences. The rules of this game obey the Markov property:

(3) $p(B(t)) = p(B(t) \mid B(t-1)$.

Maximizing (2) is thus equivalent to maximizing:

(4) $\Pi_{t=1\ldots T}\, p(\text{photo}(t) \mid B(t)) \times p(B(t) \mid B(t-1))$

To evaluate this function, p(photo(t) | B(t)) is factorized over the N independent locations, as the likelihood of stone detection, $p(\text{photo}_i(t) \mid L_i(t))$, at each L_i determined by the image patch classifier:

(5) $p(\text{photo}(t) \mid B(t)) = \Pi_{i=1\ldots N}\, p(\text{photo}_i(t) \mid L_i(t))$

Now we have the final form to maximize:

(6) $\Pi_t \{ [\, \Pi_i\, p(\text{photo}_i(t)|L_i(t)) \,] \times p(B(t)|B(t-1)) \}$.

3.5. The likely sequence is the shortest path

Consider the graph G whose nodes are all possible states at all times, $B(t)^k$ with k={all board states}, t=1…T. Include a node representing the empty board at time t=0. Allow a directed edge from state $B(t)^j$ to $B(t+1)^k$ if B^j is the result of a legal move from B^k. An edge is always added from $B(t)^k$ to $B\ (t+1)\ )^k$, the unchanged board at the next time step. The weight of an edge is the negative log of the likelihood of its destination:

(7) weight= $-\log[\, p(\text{photo}(t)|B(t)) \times p(B(t)|B(t-1)\,]$

Add a zero-weight edge from all nodes at time T to a single terminal node. Any path from the start node (the empty board at time zero) to the terminal node represents a sequence of board states S. The shortest such path maximizes (6).

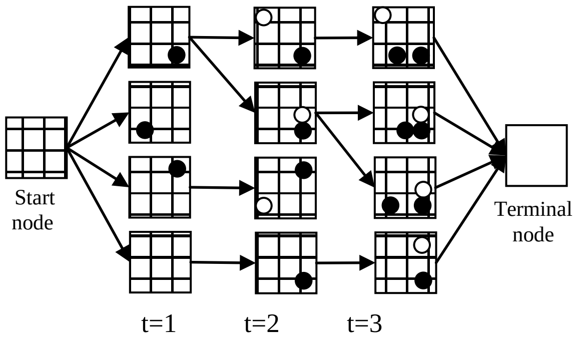

Figure 3. This Hidden Markov Model consists of many states at each time step (only a few shown), corresponding to all possible board configurations at that time step. Nonzero transition probabilities between states are sparse, corresponding to legal moves in the game (represented as arrows). One legal transition is always "no change." The most likely sequence of states is the shortest path.

3.6. Application of the A* search algorithm

The graph described above is exponentially large. For the game of Go, there are 3^{361} board states in total at each time step. Approximately 361^T of these states are reachable from the start state in T time steps. The Viterbi dynamic programming algorithm is thus inapplicable due to memory limitations. The graph must be searched implicitly, generating nodes on the fly.

The graph is searched using an augmented A* algorithm [7]. The A* algorithm requires an admissible heuristic to give a lower bound on the shortest path., which we provide by relaxing the legal-move restriction and simply taking the most likely board configuration by photo alone. We augment the standard A* algorithm by using a transposition table with Zobrist hashing [9] to avoid considering a path through a node B if a better path through that node has already been found.

The best estimate of the first move will likely be clear after only a short time. A good approximation to the single T-step problem is obtained by solving (T-D) smaller problems of D steps. For each problem, the first node of this path is greedily assumed correct. A new D-step search may then be conducted starting at that node. The results presented are achieved with D=20, corresponding to 40 seconds of real time. State transition probabilities are assumed to be a uniform distribution over legal moves and zero otherwise.

4. Results

The algorithm was tested empirically by recording several games and manually marking the true sequence of moves. The parameters are tuned to have zero or few false negatives (undetected stones). We evaluate our algorithm based on the number of corrections a user would need to make to obtain the true move sequence. A mistake is counted when the algorithm detects a nonexistent stone. The algorithm is not penalized for timing errors, only actual false positives.

Game	Mistakes in Simple classifier	Mistakes after A* correction	Number of moves in game
1	65	0	31
2	414	0	70
3	110	14	115
4	61	11	49

5. Discussion and future work

Parsing a real world scene into objects is challenging, but possible when a sufficient model of the scene is available. Board games provide well-defined scenes on the limit of recognizability, which our algorithm transcribes into semantic events.

A simple classifier operating independently at each time step produces many errors, but constraining the global time sequence of events to conform to known rules eliminates many of these mistakes.

6. References

[1] Seewald, A.K., Automatic Extraction of Go Game Positions from Images: An Application of Machine Learning to Image Mining, tech. report, Seewald Solutions, Vienna, 2007.

[2] T. Hirsimäki, GoCam: Extracting Go Game Positions from Photographs, tech. report, Teemu Hirsimäki, Helsinki University of Technology, Helsinki, 2005.

[3] K. Shiba and K. Mori, “Detection of Go-board contour in real image using genetic algorithm”, Proc Society of Instrument and Control Engineers 2004, vol. 3, Issue , 4-6 Aug. 2004, pp. 2754 -– 2759.

[4] Ball, C, Programmer, Image2SGF: Cambridge, UK: University of Cambridge, 2004. http://www.inference.phy.cam.ac.uk/cjb/image2sgf.ht ml

[5] Hartley R., A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, Cambridge, 2004.Multi View Geometry, Hartley & Zisserman

[6] W.A. Barrett and K.D. Petersen, "Houghing the Hough: Peak Collection for Detection of Corners, Junctions and Line Intersections," Proc IEEE CVPR, vol. 2, 2001, pp. 302.

[7] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pp. 97-104. Prentice Hall, Upper Saddle River, NJ, 2003.

[8] Duda, R., Hart, P., and Stork, D., Pattern Classification. John Wiley and Sons, Inc, Hoboken, NJ, 2001.

[9] Zobrist, A.” A Hashing Method with Applications for Game Playing”, Tech. Rep. 88, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, (1969)