Learning Physically-Instantiated Game Play Through Visual Observation

Andrei Barbu, Siddharth Narayanaswamy, and Jeffrey Mark Siskind

***Abstract*— We present an integrated vision and robotic system that plays, *and learns to play*, simple physically-instantiated board games that are variants of TIC TAC TOE and HEXAPAWN. We employ novel custom vision and robotic hardware designed specifically for this learning task. The game rules can be parametrically specified. Two independent computational agents alternate playing the two opponents with the shared vision and robotic hardware, using pre-specified rule sets. A third independent computational agent, sharing the same hardware, learns the game rules solely by observing the physical play, without access to the pre-specified rule set, using inductive logic programming with minimal background knowledge possessed by human children. The vision component of our integrated system reliably detects the position of the board in the image and reconstructs the game state after every move, from a single image. The robotic component reliably moves pieces both between board positions and to and from off-board positions as needed by an arbitrary parametrically-specified legal-move generator. Thus the rules of games learned solely by observing physical play can drive further physical play. We demonstrate our system learning to play six different games.**

I. INTRODUCTION

Children learn to play games by watching others play. While both formal board games, like CHESS, CHECKERS, and BACKGAMMON, and less formal play like HOPSCOTCH, TAG, and DODGEBALL all have well defined rules that children ultimately come to know, they are rarely told those rules explicitly. Knowledge of how to play many classic board games is largely passed down culturally, with children never reading, and often even explicitly ignoring, the formally-specified rules (e.g., Monopoly®). We are engaged in a long-term research effort to emulate on robots this ability to learn to play games by observing others play. The work presented here is part of a larger effort to ground learning, reasoning, and language in visual perception and motor control. Physical instantiation is crucial to our effort of situating learning, visual perception, and manipulation in the real world. We want physical robots to play a physical game where knowledge of game play allows their vision systems to determine game progress and motor systems to effect game progress. We also want a physical learner to visually observe that play to learn the game rules and ultimately be able to use the learned rules to support physical game play.

Our long-term vision for this overall task is depicted in Fig. 1. In this task, two robotic agents, the **protagonist** and **antagonist**, play a board game like CHESS. A third robotic agent, the **wannabe**, does not know the rules of the game but must infer the rules by visually observing the play of the **protagonist** and **antagonist**. The **wannabe** must then use these rules for further physically-instantiated play. In the long term, we wish to be able to do this for a wide variety of off-the-shelf game hardware for a wide variety of common physically-instantiated board games. Our objective is to learn to play *legally*, not necessarily *well*. Expert computer game play is one of the most extensively studied and successful sub-disciplines of AI. Our goal is orthogonal to that enterprise.

We have constructed a novel custom robot to support this enterprise, and have used this robot to successfully learn six different physically-instantiated games. While one long-term goal is to learn a wide variety of common board games, like CHESS, CHECKERS, BACKGAMMON, and GO, with differing physical game hardware, the work presented in this paper is limited to games which share the same game hardware. And while another long-term goal is to use three separate robotic agents to play the roles of **protagonist**, **antagonist**, and **wannabe**, the work presented here uses a single robot to play all three roles. We do however, use a unique capability of our novel custom robot to simulate play by multiple distinct agents by robotically moving the camera to image the game play from different viewpoints.

II. OUR CUSTOM ROBOT

We have designed a custom robot and built three copies thereof, one of which is shown in Fig. 2. While much of our robot is constructed with off-the-shelf parts, many crucial parts were custom designed, milled, or repurposed to meet the particular needs of the game-playing task. The two most-novel parts are the overall housing and camera-mount assembly. The overall housing consists of a two-level wood platform, where the upper level constitutes the game-play surface and the lower level serves as the mounting point for the camera assembly. A 5 DOF arm with two independently-controllable fingers, is bolted to the upper level. The size of the overall housing and the arm link lengths were designed to support game play with off-the-shelf game hardware.

Our robot hand contains a number of sensors to support fine motor control for manipulating game pieces: a palm-mounted camera, an ultrasonic range sensor, a laser pointer, and tactile force sensors on each fingertip. The camera assembly consists of a pair of pan-tilt USB webcams mounted on a 1 DOF pendulum arm which is in turn mounted on a servo base bolted to the lower level. We found Logitech QuickCam Orbit cameras well-suited to our task, as we were able to strip them down to a lightweight assembly containing the camera, pan-tilt motors, and electronics, allowing them to be mounted on the pendulum arm. This allows them to pivot,

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47907, USA {abarbu,snarayan,qobi}@purdue.edu

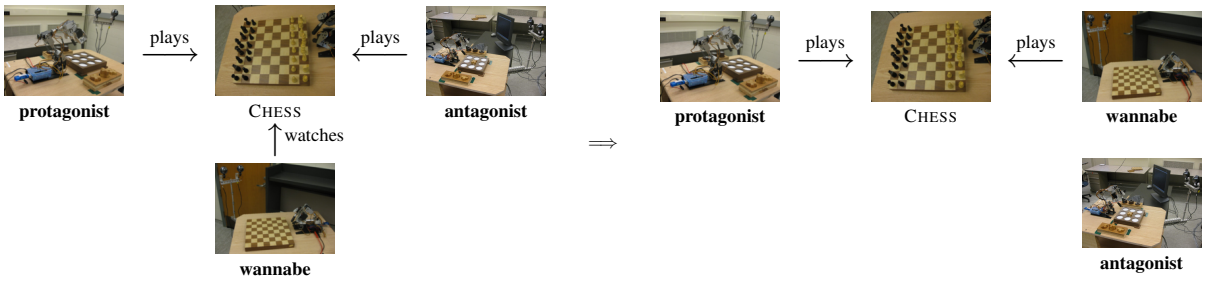

Fig. 1. Learning physically-instantiated game play through visual observation. Two robotic agents, the **protagonist** and **antagonist**, play a board game like CHESS. A third robotic agent, the **wannabe**, does not know the rules of the game but must infer the rules by visually observing the play of the **protagonist** and **antagonist**. The **wannabe** must then use these rules for further physically-instantiated play.

under computer control, 180° around the center of the game-play surface and affords a binocular view of the whole game board through the entire pivot range. We have conducted experiments both where the pendulum head mount is fixed throughout game play, imaging the game-play surface from a single viewpoint shared by the **protagonist**, **antagonist**, and **wannabe** and also where we pivot the camera under computer control to have the three agents view the game from different viewpoints.

III. THE SPACE OF GAMES CONSIDERED

For reasons to be discussed momentarily, our games share common physical game hardware consisting of an off-the-shelf TIC TAC TOE set (see Fig. 2). This particular game hardware simplifies the necessary robotic manipulation in several ways. First, the fact that the board positions are depressions makes piece placement somewhat self correcting. Second, the piece size is well matched to our manipulator. The game hardware also simplifies the process of finding board positions while reconstructing symbolic game states from visual input. In addition to the board, we have *caches* for storing off-board game pieces that are not in play. Since the off-the-shelf TIC TAC TOE set did not include such, we constructed our own that contain self-correcting circular depressions of the same size as the board.

Our long-term objective is to be able to learn any typical board game such as CHESS, CHECKERS, BACKGAMMON, GO, STRATEGO,® etc. Such a wide variety of games would require more-general perceptual and motor abilities than we have implemented. We wish to leverage our implemented perceptual and motor abilities as much as possible yet verify the generality of our overall approach by evaluating its ability to learn a variety of games. Thus we have chosen a collection of six simple games that can all be played with the same physical game hardware, robotic hardware, and perceptual and motor software. Two, TIC TAC TOE and HEXAPAWN[1] [1], are commonly-known games, while the remaining four are minor variants of HEXAPAWN. We summarize these variants below:

Variant A Non-capturing moves are forward along the diagonal instead of straight ahead.

[1]On a 3×3 board, three white pieces start on one edge and three black pieces start on the opposite edge. Pieces move and capture like CHESS pawns without *en passant* or initial two-square moves. Players win by queening and lose when unable to move.

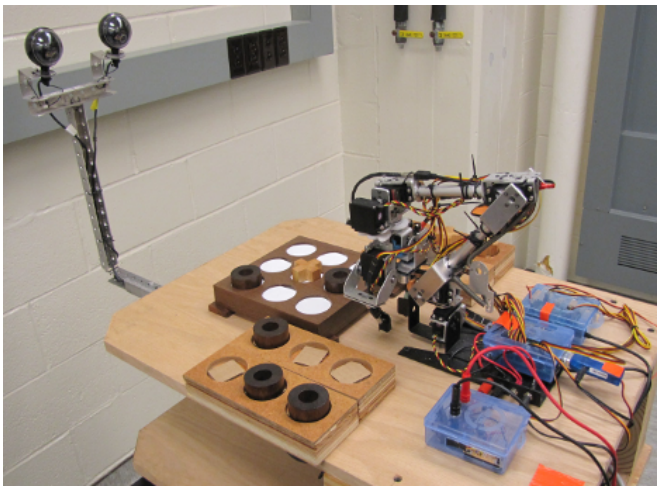

Fig. 2. One of our three novel custom robots designed specifically to support play of physically-instantiated board games. Note the 5 DOF arm with two independently-controllable fingers mounted on the upper level of a two-level housing that serves as the game-play surface. Also note the binocular pan-tilt cameras mounted on a pendulum arm that can pivot 180° around the game-play surface by being mounted on the lower level of the housing.

Variant B Variant A augmented with backward diagonal non-capturing moves.

Variant C HEXAPAWN allowing backward vertical non-capturing moves.

Variant D Variant C augmented with sideways non-capturing moves.

Our system contains a generic game-playing engine that is parameterized by a game specification containing an initial board configuration, legal-move generator, and outcome predicate. It accepts game specifications coded in either SCHEME, PROLOG (see Fig. 5), or a subset of English (see Fig. 3). Specifications in any of these three forms can be used to drive physical game play where the robot (Fig. 2) alternates play between X and O and determines the game state by visual observation of the physical game between each move, optionally moving the pendulum arm to determine the game state from different viewpoints for the different players. We provide hand-coded game specifications, in any of the forms, to the **protagonist** and **antagonist** to generate the training set for the **wannabe**, which does not have access to these hand-coded specifications. The learning component,

```
Every cache square for every player in the initial
state has some piece of that player.

A player moves by moving some piece of that player from
some cache square for that player to some empty board
square.

A player wins when every square in some row has some
piece of that player.
A player wins when every square in some column has some
piece of that player.
A player wins when every square in some diagonal has
some piece of that player.

A player draws when no player wins and that player has
no move.
```

Fig. 3. As demonstrated in the video on our website,[2] our system is able to play a physically-instantiated board game given rule specifications, such as these for TIC TAC TOE, in a subset of English.

to be described in Section VI, can learn game specifications in PROLOG for all of the above six games from visual observation of physical game play which can then be used to drive new physical game play. While our game-playing engine also supports optimal play via minimax game-tree search with alpha-beta pruning, optimal play does not facilitate learning of games rules as many situations necessary to learn the complete rule set do not arise, particularly for learning the outcome predicate in deterministic games. Thus for learning we employ random legal play.

IV. GAME-STATE RECONSTRUCTION

Physically-instantiated game play requires recovery of the game state from visual information, which nominally is a two-stage process. First, one must determine the world state, i.e., the shapes and positions of various pieces and board regions. Then, one must map this world-state information into game states. The former process is nominally a game-independent general-vision task of scene reconstruction and may incorporate camera calibration, segmentation, object recognition, metric reconstruction, and pose estimation. The latter process requires game-specific knowledge to determine which features of the reconstructed scene are relevant to the particular game being played. For example, in some games, like CHESS and CHECKERS, pieces are placed between edges, while in others, like GO, pieces are placed on edges. In the longer term, we intend to learn such game-specific knowledge of how world states map to game states along with the initial board configuration, legal-move generator, and outcome predicate. In this paper, we restrict consideration to games played with the particular game hardware discussed in Section III where all games use the same hard-coded mapping from world state to game state. Thus we conflate the two-stage process into a direct mapping from images to game states.

We employ two different game-state reconstruction methods. Our older method calibrates the reconstruction process prior to game play by determining the image regions that correspond to board positions using the ellipse detector (`cvFitEllipse`) in OPENCV [2] constrained to find a 3×3 grid. The content (empty, X, or O) of each elliptical board position during game play is determined by detecting Os as smaller ellipses and Xs as the absence of any ellipse. We make this process highly robust by sampling the ellipse-detector output at multiple thresholds and voting on the outcome. This method, however, is suitable only for a stationary head position because the ellipse detector is highly sensitive to the small changes in camera viewpoint that result when moving the head camera to a different position and back again.

Our newer game-state reconstruction method works even when the head camera moves to view the game-play surface from the perspective of different agents. Prior to game play, we calibrate the reconstruction process independently for each viewpoint by finding the image regions that correspond to board positions by dilating and thresholding the raw color image, finding connected components in the resulting binary image, and filtering the results under critical-point detection constrained to find a 3×3 grid. Critical-point detection is an iterative process on the convex hull of the centroids of the connected components, finding and removing the bottom-most three-element row of connected components from the collection of connected components in each of three iterations and recomputing the convex hull. We also calibrate color templates for the possible position states (empty, X, and O), from each viewpoint, prior to game play, by computing mean and covariance measures of HSV representations of sampled pixels from board positions with known states and classify unknown position states during game play using Mahalanobis distance.

We improve the robustness of both methods by enhancing the contrast of the circular depressions on the game board with white inserts to highlight the edges. With this, the older game-state reconstruction process is sufficiently robust that it made only two errors in the approximately 2000 reconstructions performed during the 62 games played autonomously for the experiments reported in this paper. As both occurred during the cleanup at the end of the training sequence, they did not impact the correctness of learning or subsequent physical play with the learned rules. The newer game-state reconstruction process is also sufficiently robust to complete an entire training regimen for TIC TAC TOE while varying the camera viewpoint for each agent (approximately 200 reconstructions in all) without any errors and supports learning a sound and complete game-rule specification.

While the cognitive portion of game play relies solely on the game state represented in the 3×3 grid of board positions, the robotic portion of game play requires knowing the positions of off-board pieces in the caches. We currently do not determine those positions from visual input and rely on properties of the particular robotic manipulation strategy discussed in Section V that allow inferring the positions of off-board pieces from the observed game state.

V. RULE-INDEPENDENT PIECE MANIPULATION

Physically-instantiated game play also requires robotic ability to effect the desired changes in the physical game state. This also nominally can be divided into a two-stage process. First, one must determine the necessary changes

in the world state that correspond to the desired changes in the game state, i.e., a legal move. Then, one must effect that change to the world state. The latter process is again nominally a game-independent robotic manipulation task which may incorporate forward and inverse kinematics, grasp planning, and path planning. The former process, however, requires game-specific knowledge, essentially the inverse of the game-specific knowledge needed for game-state reconstruction from visual input. Like before, we restrict consideration to games played with the particular game hardware discussed in Section III but do not conflate this into a single-stage process.

Our legal-move generator is formulated as a mapping from old game states to new game states. We formulate a generic method, particular to our class of games played on a 3×3 grid but applicable to any game in that class, that finds a minimal number of *pick up* and *put down* operations to effect the target change in the physical game state. Such operations may move pieces between two board positions, or between the caches and the board. In the case of the latter, we assume that game rules do not constrain the choice of cache location for any particular legal move and treat each cache as a last-in-first-out stack, one on one side for Xs and on the other for Os. This stack behavior is what allows indirect inference of the positions of off-board pieces from the observed game state without direct visual observation. We also have a generic "clean up" capability that can return all pieces to a state that corresponds to an arbitrary (but learned) initial board configuration. This allows completely autonomous robotic play of a sequence of games to provide training data for the learner and evaluate autonomous play with the learned rules.

The above constitutes the first stage of the two-stage process, namely mapping from target game-state changes to world-state changes. Again, we employ two different methods for the second stage, namely affecting the desired change to the world state by picking up and putting down pieces. The older method uses an open-loop dead-reckoning process. We hard-code the 3D world coordinates of the board and cache positions for our robots and employ inverse kinematics to determine a sequence of joint-angle configurations to effect a desired *pick up* or *put down* operation, parameterized by a specific board or cache position. The nature of board-game play allows straightforward planning of a collision-free path by approaching board and cache positions from above. Restriction of the game hardware to a particular piece set means that we can hard-code the grasp planning for each piece type. This is implemented by providing the *pick up* or *put down* operations with piece type as an additional parameter, derived visually. Our newer method improves upon the older method by automatically determining the 3D world coordinates of the board positions given a 3D model of the board together with visually-determined board pose. Such board-pose determination requires camera calibration, which is done automatically. Our newer method also automatically determines the parameters of the inverse kinematics via training on a fiducial. Finally, it augments the open-loop dead-reckoning process, now used only for coarse motor control, with a closed-loop visual-servoing process employing the palm camera and tactile sensors to implement the fine motor control for grasping the playing pieces.

We improve the robustness of both of the above methods by using visual feedback to confirm the success of a desired *pick up* or *put down* operation and compensating upon failure. Our combined vision and robotic-manipulation systems are sufficiently robust that the approximately 2000 *pick up* and *put down* operations during autonomous play of the 62 games for the experiments reported in this paper required fewer than 20 human interventions to correct robotic errors.

VI. LEARNING GAME RULES BY ILP

We employ inductive logic programming (ILP) with PROGOL [3] to learn perspicuous PROLOG specifications for the game rules from autonomous physically-instantiated game play. We currently do this on a single robot that plays all three roles of **protagonist**, **antagonist**, and **wannabe**, taking care not to allow information flow that could not happen if this were done on three distinct robots. The **protagonist** and **antagonist** autonomously play a sequence of random but legal games, at least six for TIC TAC TOE and at least ten for HEXAPAWN and its variants. As discussed earlier, we do not train on optimal play because this does not provide sufficient information to infer the game rules. Furthermore, games ending in a draw do not contribute to learning the outcome predicate and hence are ignored. Thus when randomly-generated training games end in a draw, additional games are played until a requisite number of non-draw games have been collected.

Each game starts with the robot autonomously setting up the physical game hardware to correspond to the initial board configuration, given an image of the current world state, which for any game in the training set but the first, contains pieces remaining on the board from the previous game. The **protagonist** and **antagonist** then alternate play by taking an image to determine the current game state (which is not stored from the previous turn), selecting a random desired next state from the legal-move generator, planning a sequence of *pick up* and *put down* operations to effect that move, and executing those operations on the robot. Independent from this, the **wannabe** takes an image between each turn to determine the sequence of game states corresponding to a game and is given the outcome of the game at each turn, which may indicate that the game is not yet over. We conduct such autonomous game play both with a stationary head camera as well as a moving head camera to image the game from different viewpoints for each agent. The only communication between the three agents is the labeling of the outcome at each turn as well as the turn-taking coordination that informs each agent of the transition time between game states.

Upon completion of the autonomous training play, the **wannabe** formulates the training set as input to PROGOL in the following form. A game state is formulated as a 3×3 matrix. Each game is formulated as a fact `initial_board(G,P)`, followed by alternating facts

`legal_move(P,G1,G2)` and `outcome(P,G,O)` for each turn, where `G` denotes a game-state matrix, `P` denotes a player, either X or O, and `O` denotes a win by either X or O. Note that we provide an outcome training fact for all moves, negated for moves that do not end in a win, which provides negative evidence for learning the outcome predicate. We then use PROGOL to learn definitions for `initial_board/2`, `legal_move/3`, and `outcome/3`. To facilitate learning, we augment the training data with the background knowledge in Fig. 4. This background knowledge is the same for all six games discussed in Section III. Much of it encodes general elementary physical and mathematical properties known by almost all children, such as arithmetic (`inc/2`, `dec/2`, `row_to_int/2`, and `col_to_int/2`), the concept of a line (`linear_test/7` and `linear_obj/3`), and the frame axiom (`replace/4`, `frame/4`, and `frame_obj/6`). Some (`player/1`, `opponent/2`, `piece/1`, `empty/1`, `owns/2`, `win_outcome/1`, `owns_outcome/2`, `owns_piece/2`, `row/1`, `col/1`, `board/1`, `ref/3`, `at/4`, and `at/5`) encode knowledge about the mapping between world state and game state which we currently do not learn but anticipate learning in the future. The remainder (`forward/3` and `sideways/2`) encode combinations of that mapping with general spatial-relation knowledge known by almost all children. To drastically reduce the learning time, we use a non-generative version of `at/4` (with cuts) while learning the legal-move generator.

During training, we first learn both the initial board configuration and the legal-move generator with the background knowledge from Fig. 4. Then, to learn the outcome predicate, we augment the training set and the background knowledge with the learned legal-move generator, along with two predicates, `has_move/2` and `has_no_move/2`, that access that learned legal-move generator. Furthermore, due to significant overlap in coverage of the clauses generated by PROGOL for the outcome predicate, we replace PROGOL's internal redundancy algorithm with one that searches for a minimal subset of the candidate clauses that covers the training set. Finally, to drastically reduce the learning time, we replace the definition of the frame axiom with one that is vacuously true while training the outcome predicate.

VII. RESULTS

Our system can learn the rules of all six games discussed in Section III from autonomous physically-instantiated play by robotic agents using the methods discussed. The learned game rules for TIC TAC TOE and HEXAPAWN are given in Fig. 5. From simulated non-robotic play, we have determined that six training examples are almost always sufficient in practice to correctly determine the game rules for TIC TAC TOE (ten for HEXAPAWN and its variants). Due to the fact that we train on random legal play, it is possible but unlikely that the training set can be pathological and contain a skewed mix of the possible game situations which can lead to incorrect generalization. Furthermore, random play can make it unlikely to observe the specific events that are necessary to learn certain aspects of the rules, such as the requirement in HEXAPAWN that players lose when unable to move. We have determined, from simulated non-robotic play, that losing in this fashion occurs with high probability among a sample of ten games but not six.

We have set up a website[2] that contains videos that demonstrate the full autonomously-played training set for each of the six games as well as subsequent autonomous play using each of the six sets of learned game rules. These video sequences were gathered with a fixed head-camera position and our older game-state-reconstruction and robotic-manipulation methods, where the **protagonist** and **antagonist** play the training set with rules specified in SCHEME and then the **protagonist** and **wannabe** play with the learned PROLOG rules (Fig. 5). The website also contains a video gathered with head-camera position varying for each of the agents and our newer game-state-reconstruction and robotic-manipulation methods, where the **protagonist** and **antagonist** play the training set with rules specified in a subset of English (Fig. 3) and then the **protagonist** and **wannabe** play with the learned PROLOG rules (Fig. 5). This website also contains the full source code for our system as well as engineering drawings for the design of our custom robot and cache hardware allowing others to replicate and build upon our work.

VIII. COMPARISON WITH RELATED WORK

On the surface, it might appear that our work resembles that of the *general game playing* community in that we both share the goal of 'learning to play' games [4], [5]. Deeper inspection, however, reveals that the apparent similarity is misleading. Our work aims to learn to play *legally*; their work aims to learn to play *well*. In more-technical terms, we take a sequence of exemplar game-play instances as input and produce a game-rule description (in the form of an initial board configuration, legal-move generator, and outcome predicate), as *output*. They take such a game-rule description as *input* and produce a strategy as output. That strategy might take the form of a static evaluator or heuristic function. As such, we are addressing complementary problems. An interesting opportunity for future work would be to cascade the two into a unified system that learns to play both legally and well from exemplar game-play instances. Indeed, such an endeavor is facilitated by the fact that the general-game-playing community has adopted a standard *game description language* (GDL) [6] for inputting the game rules to their learning systems. Fortuitously, GDL is virtually identical to the PROLOG game-rule specifications output by our system.

Our work involves three components: game-rule learning, game-state reconstruction from visual input, and robotic manipulation of board-game hardware. We know of no other work that integrates all three of these. However, there has been some work that addresses each of these components individually.

[2] `http://www.ece.purdue.edu/~qobi/icra2010`

A. Game-Rule Learning

We have found surprisingly little prior work on learning game rules from example game play. Michalski and Negri [7] employed ILP to learn a static evaluator for CHESS endgames. However, this constitutes learning to play (only part of the game) *well*, not *legally*. Levinson [8] formulated the task of learning a legal-move generator for TIC TAC TOE and HEXAPAWN as reinforcement learning, but it unfortunately does not usually converge to the correct result. Furthermore, the legal-move generator is not represented in a perspicuous human-readable format. In the above work, the game-play examples are provided symbolically, and not derived from visual input. A far as learning game rules from visual examples of game play, our efforts are most similar to those of Needham *et al.* [9]–[11] and Antanas *et al.* [12]. The former line of work processes bird's-eye view video of several simple games (PAPER SCISSORS STONE played with cards and three variants of SNAP, two played with cards and one played with dice) to produce a categorical symbolic representation of the exemplar game-state sequences and then uses ILP to learn the rules of these games. Note that the size of the game state in this work is small, $(3 \times 3 \times 2)^2 = 324$ for the card-based games and $7^2 = 49$ for dice SNAP. In contrast, the size of the game state for our games is significantly larger, 3^9. Moreover, the rules of our games are significantly more complex than theirs. Dice SNAP is specified with PROLOG rules with at most four variables and three goals. The PROLOG rules for the card-game specifications require at most seven variables and three goals. Moreover, in both of these systems, the rules contain numerous ground terms produced by the perceptual system (e.g., `rollboth`, `rollone`, `pickuplowest`, `tex0`, `tex1`, `tex2`, `pos0`, `pos1`, `pos2`), indicating tabular encoding of the rules with little conceptual generalization. In contrast, as can be seen in Fig. 5, the rules of our games have as many as fifteen variables and thirteen goals, and contain no ground terms other than the `x`, `o`, `none`, and `player_x` that appear in the initial board configuration, indicating significantly greater conceptual generalization in the legal-move generator and outcome predicate. They use essentially no background knowledge that is comparable to the background notions in our work: arithmetic, the concept of a line, spatial relations, and the frame axiom.

The above line of work, like ours, first processes the perceptual input to yield categorical symbolic descriptions that serve as the input to a purely symbolic learning process. Like ours, this pipeline is brittle in the face of errors in categorical perception. They report accuracy levels in categorical perception between 83% and 100% for dice SNAP but not those for their card games. Moreover, they report that errors in categorical perception lead to errors in game-rule learning: *We are learning from small amounts of data which is locally sparse thus a classification error may make up a significant amount of the data used to form generalisation. This may seem very fragile;*. Of the nine experiments reported (three runs for each of the three games), only one experiment yielded a sound and complete game-rule specification. In contrast, our game-state-reconstruction front end is sufficiently robust to support learning sound and complete game-rule specifications for all six of our games. Moreover, discussing the above line of work [13] Muggleton states *a key challenge will be to push the system to learn more difficult things. "It would be interesting to see if this approach will scale up to more complex games such as noughts-and-crosses [...]" he adds.* Our work does just that.

Antanas *et al.* [12] report a method for learning a subset of UNO where the categorical output of game-state reconstruction is replaced with soft evidence that is then processed by probabilistic ILP. However, they expressly disregard the softness of that evidence in the experiments that they report. Moreover, they report only approximately 95% accuracy on noise-free training data and less than 90% accuracy on noisy training data. And even when trained on noise-free data, their method produces a probabilistic game-rule specification that places 9% of the probability mass on incorrect game rules. Nonetheless, we consider this a promising approach to support robust learning of game rules despite noisy game-state reconstruction.

B. Game-State Reconstruction from Visual Input

Some prior work processes game-board images from a bird's-eye view while other work, like ours, processes game-board images from the perspective of a typical human player. Shiba and Mori [14] present a method for recovering the perimeter of a GO board from a single human-view image but do not attempt to recover piece positions or game state. Ren *et al.* [15] present a method for recovering the perimeter of a game board from a bird's-eye-view image sequence and then recover piece positions relative to this perimeter. Their method applies to an arbitrary rectangular board but they do not report using piece position to reconstruct game state and do not report accuracy. Hirsimäki [16] presents a method for recovering GO-board states from human-view images. However, they only report testing their method on six images, out of which it only succeeds in recovering the correct game state on four. Kang *et al.* [17] present a method for recovering a sequence of GO-game moves from bird's-eye-view image sequences. However, they do not report any quantitative evaluation of game-state reconstruction accuracy. Torre *et al.* [18] present a method for recovering a sequence of CHECKER-game moves from human-view image sequences of games played on a game board with a known geometric model. However, they report neither qualitative nor quantitative assessment of game-state reconstruction accuracy. Scher *et al.* [19] present a method for recovering a sequence of GO-game moves from human-view image sequences. A novel aspect of this work is that they improve upon the purely vision-based game-state reconstruction by incorporating the rules of GO as constraints. Without such constraint, their reconstruction accuracy is extremely poor: a total of 650 piece classification errors out of 265 moves. Even adding the game-rule constraints only reduces the number of piece classification errors on this data set to 25. Note that these

numbers constitute errors in recognizing the presence or absence of a black or white stone at individual positions, not the aggregate whole-game-state recognition accuracy, which would be significantly lower. Seewald [20] presents a method for recovering game states from human-view images of GO boards with a reported whole-game-state recognition accuracy of 72.7%. In contrast, we recover game moves from human-view image sequences with only two erroneously reconstructed game states out of approximately 2000. We are unaware of any prior work that achieves anywhere near this level of accuracy. Such performance is crucial for accurately learning game rules. Moreover, we cannot avail ourselves of the technique employed by Scher *et al.* as the **wannabe** does not yet know the game rules.

C. Robotic Manipulation of Board-Game Hardware

The robotic ability to manipulate board-game pieces by dead reckoning is straightforward and commonplace, and there have been attempts to integrate such ability with automated board-game play based on visual perception of game states and piece positions [21]–[23]. However, we are unaware of any prior work that reports such a combined sensorimotor game-play system that achieves the level of robustness needed for fully-automated play of a training set that is sufficiently large to support game-rule learning. Indeed, despite the fact that they use a bird's-eye-view camera together with a vacuum grip to pick up TRAX tiles, alleviating the need for fine motor control, Bailey *et al.* [22] state *At present, the robot is a little clumsy, frequently placing tiles slightly over the top of other tiles.*

Furthermore, an important aspect of our work is the way our robot was designed specifically to support game-rule learning. While our robot uses a common 5 DOF arm, this arm is mounted in a novel environment. We know of no prior robot that incorporates our two-level housing design and its associated pendulum head mount. Moreover, we know of no prior attempt to model multi-agent robotic board-game-play systems by automatically changing the camera view by such robotic means to model each agent. Our combined vision and robotic system is sufficiently robust not only to support game-state reconstruction and game-piece manipulation from varying camera views, it can do so even with the imaging variation due to inherent inaccuracy in repeated attempts to move the head camera to the same position.

IX. CONCLUSION

We have presented the first integration of vision, robotics, and game-rule learning in the realm of board-game play. Our work is also a significant advance over prior independent work in each of these areas. The richness of board-game play allows for an open-ended research program investigating perceptual and motor grounding of natural language, reasoning, and learning yet its circumscribed nature allows for robust incremental progress.

X. ACKNOWLEDGMENTS

This work was supported, in part, by NSF grant CCF-0438806. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

REFERENCES

[1] M. Gardner, "Mathematical games," *Scientific American*, Mar. 1962.
[2] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Press, Oct. 2008.
[3] S. Muggleton, "Inverse entailment and Progol," *New Generation Computing, Special issue on Inductive Logic Programming*, vol. 13, no. 3-4, pp. 245–286, 1995.
[4] S. Schiffel and M. Thielscher, "Fluxplayer: a successful general game player," in *Proceedings of the 22nd National Conference on Artificial intelligence*, 2007, pp. 1191–1196.
[5] J. Clune, "Heuristic evaluation functions for general game playing," in *Proceedings of the 22nd National Conference on Artificial intelligence*, 2007, pp. 1134–1139.
[6] M. Genesereth and N. Love, "General game playing: Game description language specification," Computer Science Department, Stanford University, Stanford, CA, USA, Tech. Rep., 2005.
[7] R. S. Michalski and P. Negri, "An experiment on inductive learning in chess end games," in *Machine Intelligence 8*, E. Elcock and D. Michie, Eds. New York: Horwood, 1977, pp. 175–192.
[8] R. Levinson, "General game-playing and reinforcement learning," University of California at Santa Cruz, Santa Cruz, CA, USA, Tech. Rep., 1995.
[9] D. R. Magee, C. J. Needham, P. E. Santos, A. G. Cohn, and D. C. Hogg, "Autonomous learning for a cognitive agent using continuous models and inductive logic programming from audio-visual input," in *Proceedings of the AAAI Workshop on Anchoring Symbols to Sensor Data*, 2004.
[10] C. J. Needham, P. E. Santos, D. R. Magee, V. Devin, D. C. Hogg, and A. G. Cohn, "Protocols from perceptual observations," *Artificial Intelligence*, vol. 167, pp. 103–136, 2005.
[11] P. E. Santos, S. Colton, and D. R. Magee, "Predictive and descriptive approaches to learning game rules from vision data," in *IBERAMIA-SBIA*, 2006, pp. 349–359.
[12] L.-A. Antanas, I. Thon, M. van Otterlo, N. Landwehr, and L. De Raedt, "Probabilistic logical sequence learning for video," in *Inductive Logic Programming*, Jul. 2009.
[13] W. Knight, "Machine learns games 'like a human'," *New Scientist*, 2005. [Online]. Available: http://www.newscientist.com/article/dn6914
[14] K. Shiba and K. Mori, "Detection of Go-board contour in real image using genetic algorithm," in *SICE Annual Conference*, vol. 3, Aug. 2004, pp. 2754–2759.
[15] J. R. Peter, P. Astheimer, and I. Marshall, "A general framework for vision based interactive board games," in *4th Annual European GAME-ON Conference*, 2003.
[16] T. Hirsimäki, "Gocam: Extracting Go game positions from photographs," Helsinki University of Technology, Helsinki, Finland, Tech. Rep., 2005.
[17] D. C. Kang, H. J. Kim, and K. H. Jung, "Automatic extraction of game record from TV Baduk program," in *The 7th International Conference on Advanced Communication Technology*, vol. 2, 2005, pp. 1185–1188.
[18] R. Torre, P. Fua, S. Balcisoy, M. Ponder, and D. Thalmann, "Interaction between real and virtual humans: Playing Checkers," in *Proc. Eurographics Workshop on Virtual Environments*, 2000.
[19] S. Scher, R. Crabb, and J. Davis, "Making real games virtual: Tracking board game pieces," in *19th International Conference on Pattern Recognition*, Dec. 2008, pp. 1–4.
[20] A. Seewald, "Automatic extraction of Go game positions from images: An application of machine learning to image mining," Seewald Solutions, Tech. Rep., 2007.
[21] B. Marsh, C. Brown, T. LeBlanc, M. Scott, T. Becker, C. Quiroz, P. Das, and J. Karlsson, "The Rochester Checkers player: Multimodel parallel programming for animate vision," *Computer*, vol. 25, no. 2, pp. 12–19, 1992.
[22] D. G. Bailey, K. A. Mercer, and C. Plaw, "Autonomous game playing robot," in *2nd International Conference on Autonomous Robots and Agents*, 2004.
[23] José Gonçalves and José Lima and Paulo Leitõ, "Chess robot system: A multi-disciplinary experience in automation," in *9th Spanish Portuguese Congress On Electrical Engineering*, 2005.

```
inc(X,Y):-Y is X+1.

dec(X,Y):-Y is X-1.

player(player_x).
player(player_o).

opponent(player_x,player_o).
opponent(player_o,player_x).

piece(x).
piece(o).
piece(none).

empty(none).

owns(player_x,x).
owns(player_o,o).

win_outcome(x_wins).
win_outcome(o_wins).

owns_outcome(player_x,x_wins).
owns_outcome(player_o,o_wins).

owns_piece(x_wins,x).
owns_piece(o_wins,o).

row(r0).
row(r1).
row(r2).

col(c0).
col(c1).
col(c2).

row_to_int(r0,0).
row_to_int(r1,1).
row_to_int(r2,2).
```

```
col_to_int(c0,0).
col_to_int(c1,1).
col_to_int(c2,2).

board([[A,B,C],[D,E,F],[G,H,I]]):-
        piece(A),piece(B),piece(C),
        piece(D),piece(E),piece(F),
        piece(G),piece(H),piece(I).

ref(0,[H|_],H).
ref(0,_,_):-!,fail.
ref(1,[_,B,_],B).
ref(1,_,_):-!,fail.
ref(2,[_,_,C],C).
ref(2,_,_):-!,fail.
ref(I,[H|T],H2):-dec(I,J),ref(J,T,H2).

at(ROW,COLUMN,BOARD,PIECE):-
        row_to_int(ROW,RI),
        col_to_int(COLUMN,CI),!,
        ref(RI,BOARD,L),!,ref(CI,L,PIECE).
at(ROW,COLUMN,BOARD,PIECE,
   [ROW,COLUMN,PIECE]):-
        at(ROW,COLUMN,BOARD,PIECE).

replace(0,A,[_|T],[A|T]):-!.
replace(0,_,_,_):-!,fail.
replace(1,A,[X,_|T],[X,A|T]):-!.
replace(1,_,_,_):-!,fail.
replace(2,A,[X,Y,_|T],[X,Y,A|T]):-!.
replace(2,_,_,_):-!,fail.
replace(I,H1,[H2|T1],[H2|T2]):-
        dec(I,J),replace(J,H1,T1,T2).

frame(RROW,CCOLUMN,BOARD1,BOARD2):-
        row(RROW),
        col(CCOLUMN),
        row_to_int(RROW,ROW),
        col_to_int(CCOLUMN,COLUMN),
        ref(ROW,BOARD1,L1),
        replace(ROW,L2,BOARD1,BOARD3),
        replace(COLUMN,ignore,L1,L2),
        ref(ROW,BOARD2,L3),
        replace(ROW,L4,BOARD2,BOARD3),
        replace(COLUMN,ignore,L3,L4),!,
        board(BOARD2).
```

```
frame_obj([R1,C1,P1],
          [R2,C2,P2],
          [R1,C1,P3],
          [R2,C2,P4],
          B1,
          B2):-
        frame(R1,C1,B1,B3),
        at(R1,C1,B1,P1),
        at(R2,C2,B1,P2),
        frame(R2,C2,B3,B2),
        at(R1,C1,B2,P3),
        at(R2,C2,B2,P4).

forward(player_x,R1,R2):-
        row_to_int(R1,RI1),
        inc(RI1,RI2),
        row_to_int(R2,RI2).
forward(player_o,R1,R2):-
        row_to_int(R1,RI1),
        dec(RI1,RI2),
        row_to_int(R2,RI2).

sideways(C1,C2):-
        col_to_int(C1,CI1),
        inc(CI1,CI2),
        col_to_int(C2,CI2).
sideways(C1,C2):-
        col_to_int(C1,CI1),
        dec(CI1,CI2),
        col_to_int(C2,CI2).

linear_test(X1,Y1,X2,Y2,X3,Y3,S):-
        S is X1*(Y2-Y3)+X2*(Y3-Y1)+
               X3*(Y1-Y2).

linear(R1,C1,R2,C2,R3,C3):-
        [R1,C1]\=[R2,C2],
        [R3,C3]\=[R2,C2],
        [R1,C1]\=[R3,C3],
        row_to_int(R1,RI1),
        row_to_int(R2,RI2),
        row_to_int(R3,RI3),
        col_to_int(C1,CI1),
        col_to_int(C2,CI2),
        col_to_int(C3,CI3),
        linear_test(RI1,CI1,RI2,
                    CI2,RI3,CI3,0).
```

Fig. 4. Background knowledge encoded in PROLOG. PROGOL-specific settings and mode, type, and pruning declarations have been omitted.

TIC TAC TOE

```
initial_board([[none,none,none],
               [none,none,none],
               [none,none,none]],
              player_x).
legal_move(A,B,C) :- owns(A,D),
        row(E),
        col(F),
        at(E,F,B,none,G),
        at(E,F,C,D,H),
        frame_obj(G,H,G,H,B,C).
outcome(A,B,C) :- owns_piece(C,D),
        at(E,F,B,D),
        at(G,H,B,D),
        at(I,J,B,D),
        linear(E,F,G,H,I,J).
```

HEXAPAWN

```
initial_board([[x,x,x],
               [none,none,none],
               [o,o,o]],
              player_x).
legal_move(A,B,C) :- row(D),
        col(E),
        owns(A,F),
        empty(G),
        forward(A,H,D),
        at(H,E,B,F,I),
        at(H,E,C,G,J),
        at(D,E,B,G,K),
        at(D,E,C,F,L),
        frame_obj(I,K,J,L,B,C).
```

```
legal_move(A,B,C) :- row(D),
        col(E), opponent(A,F),
        owns(A,G), empty(H),
        forward(A,I,D),
        owns(F,J),
        sideways(E,K),
        at(D,K,C,G,L),
        at(I,E,B,G,M),
        at(I,E,C,H,N),
        at(D,K,B,J,O),
        frame_obj(L,N,O,M,C,B).
outcome(A,B,C) :- row(D),
        opponent(A,E),
        forward(E,D,F),
        forward(E,F,G),
        owns_outcome(E,C),
        owns_piece(C,H),
        at(G,I,B,H,J).
outcome(A,B,C) :- opponent(A,D),
        has_no_move(A,B),
        owns_outcome(D,C).
```

Fig. 5. Rules for two of the six games discussed in Section III learned automatically from visual observation of autonomous physically-instantiated game play.