Gambit: An Autonomous Chess-Playing Robotic System

Cynthia Matuszek, Brian Mayton, Roberto Aimi, Marc Peter Deisenroth, Liefeng Bo,
Robert Chu, Mike Kung, Louis LeGrand, Joshua R. Smith, Dieter Fox

***Abstract*—This paper presents Gambit, a custom, mid-cost 6-DoF robot manipulator system that can play physical board games against human opponents in non-idealized environments. Historically, unconstrained robotic manipulation in board games has often proven to be more challenging than the underlying game reasoning, making it an ideal testbed for small-scale manipulation. The Gambit system includes a low-cost Kinect-style visual sensor, a custom manipulator, and state-of-the-art learning algorithms for automatic detection and recognition of the board and objects on it. As a use-case, we describe playing chess quickly and accurately with arbitrary, uninstrumented boards and pieces, demonstrating that Gambit's engineering and design represent a new state-of-the-art in fast, robust tabletop manipulation.**

***Index Terms*—Mechanism Design of Manipulators; Physical Human Robot Interaction**

I. INTRODUCTION AND RELATED WORK

Physical board games are a rich problem domain for human-robot cooperation research because such games have an intermediate and easily adjustable degree of structure. Playing board games involves perception of the board and game pieces, perception of the human, reasoning about the game and game state, and manipulation of the physical pieces while coordinating with the human opponent. Progress on physical board game playing systems paves the way for more general human-robot cooperation systems that assume less structure. For example, this line of work could lead eventually to a manipulator capable of helping a chemist as a lab assistant that cooperatively performs manipulation tasks in an unstructured laboratory bench-top or "glove box" environment.

This paper introduces *Gambit*, a robot manipulator system that is designed to autonomously play board games against human (or robotic) opponents. In this paper, we focus on the game of chess, see Fig. 1. A large number of chess playing automata that have been imagined or constructed in the last three centuries suggests that robot chess could be interesting as an entertainment application; instead, we view robot chess primarily as a testbed problem of adjustable difficulty that can advance research in perception and manipulation in a noisy, less constrained real-world environment.

Compared to prior work on robotic systems playing chess with specifically instrumented chess boards and/or pieces, the Gambit system represents a leap forward in generality, and naturally suggests several future steps in the direction of decreasing structure and increasing generality.

Fig. 1. Gambit autonomously plays chess against a child.

Gambit can play with arbitrary chess sets on a variety of boards, requiring no instrumentation or modeling of pieces. Gambit monitors the board state continuously and detects when and what kind of move an opponent has made. Furthermore, Gambit communicates with a human opponent through a natural spoken-language interface.

We developed a custom robotic arm hardware for the Gambit system to create an open, flexible platform that supports future exploration of robot hardware cost scaling—for example, techniques for shifting performance and cost from mechanical components to sensors, silicon, and software.

Gambit's perceptual system tracks the board pose and the human opponent in real time. The board is not fixed relative to the robot and is continuously calibrated during game play. This allows for game play without an explicit indication of move completion.

The Mechanical Turk, first exhibited in 1770, is perhaps the first purported chess playing automaton. It was in fact not autonomous at all, as it relied on a concealed human chess master for perception, game logic, and control of the Turk's manipulation hardware, which consisted of a mechanical arm and hand, as well as a "voice box" that could say a single word ("Echec"). The Turk used magnetically instrumented chess pieces that enabled the operator to sense the game state via the motion of corresponding magnets in the operator's compartment. Thus the mechanical Turk was actually a chess teleoperation system, not a chess automaton [17], [19].

Nearly all autonomous board game playing systems use instrumented or special purpose game boards and pieces to simplify perception and manipulation. Thus, they do not handle the complexities introduced by using arbitrary pieces,

C. Matuszek, M. P. Deisenroth, L. Bo, R. Chu, M. Kung, J. R. Smith, and D. Fox are with the Dept. of Computer Science & Engineering, University of Washington, Seattle, WA; B. Mayton is with the MIT Media Lab, Cambridge, MA; R. Aimi is with Alium Labs LLC; L. LeGrand and D. Fox are with Intel Labs Seattle, Seattle, WA; J. R. Smith is with the Dept. of Electrical Engineering, University of Washington, Seattle, WA.

This work was funded in part by an Intel grant, by ONR MURI grants N00014-07-1-0749 and N00014-09-1-1052, by the NSF under contract IIS-0812671, and through the Robotics Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement W911NF-10-2-0016.

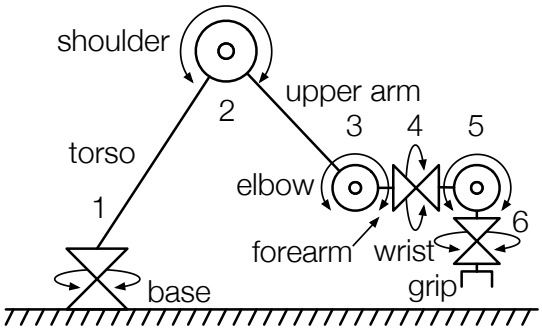


Fig. 2. Schematic of the arm degrees of freedom.

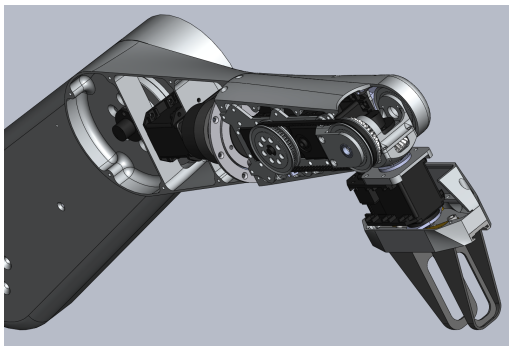


Fig. 3. Cross sectional view of Gambit forearm with covers removed.

boards, and environments. For example, chess playing robot arms were shown in 2010 at the Maker Faire,[1] and at several press events in Russia [5]. They used instrumented chess sets such as the Digital Game Technology Sensory Chess Board[2] to eliminate the perception problem. A commercial product called the Novag 2 Chess Computer includes a robot arm that can play against a person. However, it uses a special instrumented board for perception and special chess pieces that are co-designed with the manipulator.[3]

In the human-robot interaction literature, turn taking has been studied extensively in several contexts, including social speech interactions [7], drumming [13], and game play for autism therapy [9]. Human-robot collaboration and joint activity has been discussed in [6], which investigates socially expressive and natural communication via expressive robots rather than board game playing and/or co-manipulation.

The paper is organized as follows. In Sec. II, we describe Gambit's hardware and sensor design. Sec. III describes our approach to playing chess in a flexible, robust manner against human opponents. Sec. IV discusses experimental results.

II. MANIPULATOR AND SENSOR SUITE DESIGN

The Gambit system includes a completely new arm, the design of which is open source. The decision to create a new arm design was driven by several factors: First, for robotic manipulation to have widespread impact, the cost of manipulation hardware must decrease. The Gambit arm is moderate in cost (circa $18K in parts) and relatively high in precision; our hope is to seed a community-wide process of cost reduction engineering by open-sourcing the design.

Second, to be able to implement smooth, natural, sensor-driven motions and interactions (such as visual- or E-Field-servoing [15]), or to implement programmable compliance, we wanted complete control of the robot, down to low-level firmware and shortest timescales.

A. Mechanical Design

The Gambit manipulator consists of a 6-DoF arm with a parallel jaw gripper. The arm's DoFs are illustrated schematically in Fig. 2. The three revolute DoFs (1, 2, and 3) provide position control, while DoFs 4, 5, and 6 provide orientation control via a series roll-pitch-roll spherical wrist. The "torso," the structure between the base and shoulder, is offset from the base rotation axis; this torso "lean" reduces the torque requirements of the shoulder joint. The tabletop working area is a circle with a radius of $\approx 60\,\mathrm{cm}$.

To construct a 6-DoF robot with reasonable cost, we prioritized precision of position over orientation. The three position DoFs use Harmonic Drive FHA series integrated motors, zero backlash gearboxes, and 0.0018-degree encoders, driven by Copley Controls servo amplifiers. These actuators are hollow and use double needle bearings to support significant cantilever loads, simplifying joint design and cabling. The wrist uses three RX-28 Dynamixel actuators with a much lower resolution of 0.29 degrees. The Dynamixels have an integrated controller, yielding a much smaller overall package.

To achieve a spherical wrist, DoFs 5 and 6 use timing belts to offset the servo motor from the DoF, as illustrated in Fig. 3. DoF 6 uses a single hollow miter gear pair to achieve a 90-degree turn. In order to hold DoF 6 still while rotating DoF 5, both motors need to rotate. A differential design was considered but rejected because of limited space for cabling and slip ring. Slip rings are located on all necessary joints in order to allow continuous rotation.

Twelve conductors are wired to the gripper, four for a USB camera centered in the gripper, four for the gripper servo, and four are reserved for future use. Gambit uses a simple parallel jaw gripper design with a Dynamixel RX-10 in a double crank mechanism. The gripper jaws are easily replaceable to adapt to specialized tasks, and the entire gripper assembly can be retrofitted if another type of gripper is required. For picking up chess pieces, we used a hollow gripper jaw design with a rubber finger tip (normally used to aid people in collating papers) stretched over the frame, forming a compliant "opposing trampoline" structure that conforms to objects and has a slight centering effect.

The structure of the robot is milled from aluminum. The Copley Controls drivers use the structure as a heat sink, eliminating the need for a cooling fan.

B. Sensing

In addition to joint encoders, Gambit has a shoulder-mounted PrimeSense (`http://www.primesense.com/`)

[1] `http://www.chessplayingrobot.com`
[2] `http://www.digitalgametechnology.com`
[3] `http://tiny.cc/novag-2-chess-robot`

depth camera (technologically identical to an Xbox Kinect), and a small camera built into the gripper. The PrimeSense camera provides three color channels (RGB) plus depth for each pixel, and has a working range of approximately $0.5\,\mathrm{m}$ to $5\,\mathrm{m}$. This large minimum range presented a design challenge for the system. Since our goal was to develop a standalone, integrated system, we wanted the camera to be mounted on the arm rather than overhead. We solved this by mounting the depth camera facing backwards on the torso to increase camera distance to the workspace. The arm is positioned backwards with the torso leaning away from the board when perceiving board state. In normal operation, the torso leans forward which maximizes the robot's reach.

The small camera in Gambit's gripper was originally designed to fit in the bezel of an Apple MacBook. The challenge in integrating this palm camera was to maintain the integrity of its high-speed USB data as it passes through all five of Gambit's slip rings and the electrically noisy environment inside the arm. We used a USB hub in the robot's forearm to boost the signal after it passed through the two wrist slip rings, and used shielded cable wherever possible.

C. Driver Software

The driver software for the Gambit arm is arranged hierarchically, targeting specific hardware at the low levels and providing broader abstraction at higher levels; thus, the drivers above the lowest layer are reusable for other robot arms.

As explained in Sec. II-A, Gambit uses two different kinds of actuators. DoFs 1, 2, and 3 are addressed via CAN bus. DoFs 4, 5, and 6 use an RS-485 bus. The low level `gambit_driver`, which is built on top of ROS [16], provides a uniform interface to the heterogeneous actuators.

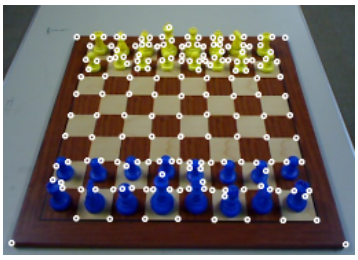
The drivers run on a dedicated Intel Atom net-top PC equipped with CAN and RS-485 PCI cards. This dedicated control PC is not subject to variable load conditions that might interfere with smooth control of the arm. Applications running on a separate computer command the arm using a higher-level driver that provides an `Arm` object, which handles ROS communication with the lower level drivers.

III. Perception and Manipulation

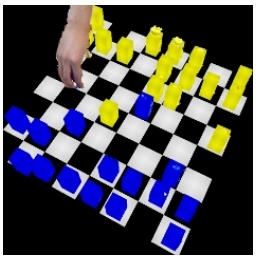
In the following, we describe details about the perception and manipulation required for autonomous chess playing. This includes real-time tracking of the location and the board state and detection of an opponent's move, see Sec. III-A. Sec. III-B describes how Gambit learns to recognize chess pieces, a first step toward Gambit joining a chess game at any stage of the game. In Sec. III-C, we describe the actual process of playing a game, including details about the manipulation and the interface Gambit uses to communicate with a human.

A. Perception and Game State Estimation

1) Locating the Chessboard: Locating the chessboard and continuously updating its posture with respect to the camera involves: (1), finding the board itself, and (2), finding the transformation of the board with respect to the camera. Ignoring the depth information from the camera (Sec. II-B), we detect corner points on the 2D (RGB) image of the chess board to locate the board and the grid of squares, see Fig. 4(a). Depth information of the corner points is then incorporated. A plane is fitted to the (now 3D) points using RANSAC [12], which yields the plane upon which the board surface lies, but does not uniquely determine the exact board placement. Thus, we determine the best match of all 3D points on the board plane with a template of the 8×8 contiguous chessboard cells to localize the board. This approach is also robust to partial occlusions of corner points (e.g., by a hand or other pieces).



(a) Board corner points in white.



(b) Detection of a hand while moving a chess piece.

Fig. 4. (a): Finding 2D corner points. (b): Hand detection.

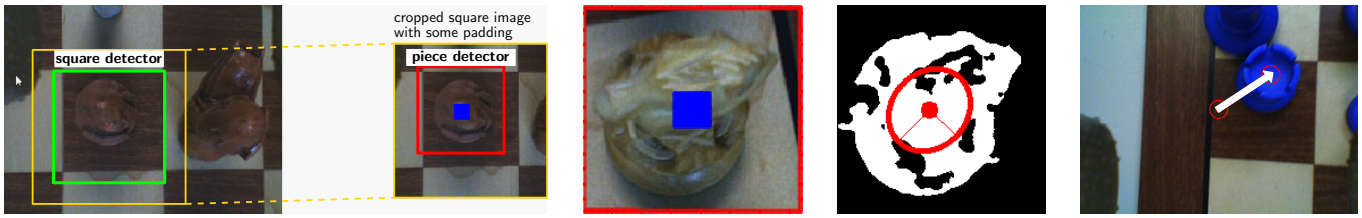
2) Board Occupancy: To determine if chess squares are occupied, Gambit uses the point cloud consisting of all points above the surface of the board plane. An isometric projection of this point cloud onto the board returns a 2D projection of all pieces onto the board plane. The projected points are clustered, and individual clusters—corresponding to pieces—are assigned to a cell in the 8×8 *occupancy grid* defining the chess board. When an object, e.g., a hand, is detected in the area above the board, see Fig. 4(b), the determination of the occupancy grid is paused until the object disappears.

Piece *color* at occupied squares is assigned in two steps: At the beginning of the game, we define "black" and "white" by computing the 2-median colors, c, for the point clouds corresponding to all pieces on either side of the chessboard. During the game, Gambit determines the best assignment of the clustered projected point clouds to "black"/"white". This procedure returns the current occupancy of the board and the colors of the occupying pieces at $15\,\mathrm{Hz}$.

To track the piece *types* on the board, we first define the *board state* as a tupel $(s_i, c_i, p_i)_t$, $i = 1, \ldots, 64$, where $s \in \{0, 1\}$ is a binary square occupancy variable, $c \in \{\text{white}, \text{black}\}$ is the chess piece color, t describes the time step, and $p \in \{\text{K, Q, P, R, B, N}\}$ for **K**ing, **Q**ueen, **P**awn, **R**ook, **B**ishop, and k**N**ight, respectively, defines the piece type. Second, the board occupancy and color differences of two consecutive board states are computed. GNU chess uses these differences to check the validity of the change. If the change is invalid, the opponent is asked for correction.

B. Chess Piece Detection and Recognition

In the long term, Gambit is supposed to join a chess game at any stage. Currently, Gambit needs a known initial configuration to identify the chess pieces. As a necessary step



(a) From a palm-camera image to a detected chess piece. (b) SVM training set. (c) Principal axes for rotational correction. (d) Translational correction.

Fig. 5. From a full image to rotational and translational corrections of the end effector illustrated across all considered chess sets. (a): Palm-camera image, detected square, and detected chess piece. (b): Cropped image around the detected chess piece (enlarged box from (a), but using a different piece type). Positive (blue, center) and negative SVM training labels (red, image border). (c): the corresponding segmented image used for visual servoing, the center of the blob and its two principal axes (red ellipse) from which the rotational correction is computed. (d): Global translational correction.

toward our long-term goal, we now present a piece recognition system that can be used to identify chess pieces.

Our chess-piece detection and recognition system consists of four hierarchical classifiers: a square detector, a binary piece/background detector, a binary piece color (white/black) recognizer, and two piece-type recognizers, each with the six class labels {B,N,K,P,Q,R}. Fig. 6 details the main steps for chess-piece recognition during game play given a 640×480 palm-camera image. In the following, we detail the classifiers.

1) Square Detector: The square detector is used to find chess squares—independent of rotation and scale—in the 640×480 palm-camera image.

To be robust to rotations of the camera, we generate 20 square templates by rotating a 220×220 pixel square. From the templates, we extract histograms of oriented gradients (HOG) features [8] and obtain 20 HOG square *training* templates. Using the HOG square templates turns out to be more robust than using Hough transformation, which only returns lines, but not squares. During *testing*, we follow a standard sliding window approach, evaluate a score function for all positions in an image, and threshold the scores to obtain bounding boxes for the squares. The left panel in Fig. 5(a) shows an example of a detected square in the full image.

For robustness to scaling of the squares, due to different chess boards or different hover heights of the robot arm, an image scale is chosen at *test time* to ensure that the square size in the image corresponds approximately to the size of square training templates. To find an appropriate scale, we generate an image pyramid using ten pre-specified scales $0.8\times(1.05)^i$, $i = 1,\ldots,10$. We apply the square detector to the image pyramid and find the best scale. We repeat this procedure 20 times and choose the overall-most likely scale.

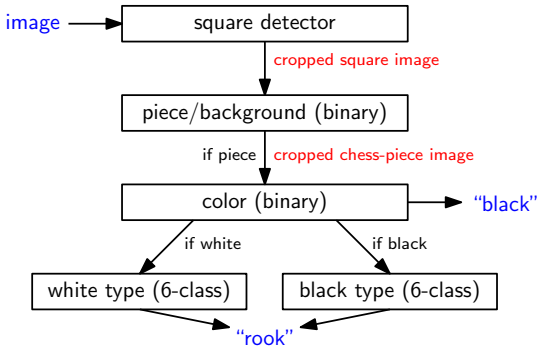


Fig. 6. Classifier hierarchy for chess-piece recognition during game play.

2) Piece/Background Detector: For training the chess-piece detector, padded cropped images from the square detector (240×240–300×300 pixels) with chess pieces in their centers serve as positive examples (see right panel in Fig. 5(a)), equally-sized background images serve as negative examples. On this training set, we train a binary linear SVM using LIBLinear [10], which defines our piece/non-piece detector. We padded the detected chess squares (marked orange in Fig. 5(a)) to account for large chess pieces and/or camera tilt.

The training examples are generated automatically using the square detector described previously: The square detector typically yields up to three squares in the 640×480 image. We choose the square closest to the center of the image: Optimally, the robot arm and the palm camera would be centered exactly at the target square). The left panel in Fig. 5(a) shows the detected square closest to the image center. Finally, a rectangular image window centered at the target square is the training example for the SVM (see Fig. 5(a), right panel).

3) Color Detector: The chess pieces have two colors ("black"/"white"). We take the chess-piece bounding box, see Fig. 5(b), to train a color classifier. For example, in Fig. 5(b), the blue pixels are training data for "white" chess pieces.

4) Chess Piece Classifier: As illustrated in Fig. 6, for each color, a chess piece classifier is trained to distinguish the piece types. The features of the respective chess-piece classifiers are concatenated SIFT [14] and kernel descriptors [3]. The features are extracted with 16×16 image patches over dense regular grids with spacing of 8 pixels. We use these local features to compute efficient match kernel (EMK) features [4] to obtain the final image level feature.

C. Game Playing and Manipulation

We consider playing a non-simplified chess game from a known initial configuration of the board: Gambit is capable of making any legal chess move, including castling and piece capture. Furthermore, Gambit can identify these moves when made by an opponent. See [1] for videos of examples.

At the beginning of the game, the system builds an initial occupancy grid, computes and saves models of piece color, and stores a table of the heights, obtained from the depth

camera, of different piece types. This information is used through the rest of the game. Once the occupancy grid has been constructed, the current state of the board is tracked using depth and color information from the PrimeSense camera.

Playing is thereafter a cycle of perceiving the board state, checking for problems, deciding on and making a move, and checking again. Examples of problems that can be detected include a failure in making a move, an opponent making an illegal move, or a piece becoming occluded by another piece. In these cases, Gambit asks for help and game play is paused.

A complete move (from the end of an opponent placing a piece to the end of Gambit's subsequent move) averages 22.5 seconds, which is well within human turn-taking time.

1) User Interface: Gambit communicates both the game state and encountered problems via a natural-language spoken interface. Every move taken during the game is repeated in spoken colloquial language. When an error is detected—either on the robot's part (e.g., dropping a piece), or on the human's part (e.g., making an illegal move)—Gambit verbally requests human intervention to make it possible to continue.

2) Chess Piece Manipulation: Depending on the move, a sequence of manipulations is required. For example, when to capture an opponent's piece, Gambit first removes the opponent's piece from the board and then moves the capturing piece to the desired location. Picking up a piece requires moving the end effector above the center of the board square containing the piece, optionally using visual servoing for local adjustments, lowering the end effector to a piece-dependent height, and closing the gripper. Depositing a piece is similar.

3) Visual Servoing for Local End Effector Adjustment: For asymmetric chess pieces or pieces not centered on a chess square, standard grasps can fail. To improve grasp success, Gambit performs visual servoing for local corrections. Visual servoing is based on images from the palm camera.

As in [11], low-dimensional image features obtained from the palm-camera image are used to correct both the 2D pose of the end effector in the hovering plane above the board and the roll angle of the manipulator. These features—the center and the orientation of the chess piece—are computed from the statistics of a binary image, which itself uses color and edge-based segmentation. Specifically, the local adjustments are computed with the loop:

1) Obtain a cropped image with a single chess piece from the square and piece detectors in Sec. III-B (Fig. 5(a), right panel).
2) To deal with changing lighting conditions, we use *online piece segmentation* using color and edge information: We distinguish between the classes "chess piece" and "background" using a kernel SVM [18]. Positive training inputs for the SVM are the pixels in a 32×32 square in the center of the image, negative inputs are the same number of points along the image border, see Fig. 5(b). After training, the entire cropped image serves as the test input. The SVM returns a binary class label per pixel (Fig. 5(c)).

Fig. 7. Gambit can play using arbitrary chess sets. Shown are yellow and blue Grandmaster pieces (lower left), reproduction Lewis Chessmen pieces (lower right), and Magic Ball pieces (back row).

3) Compute the center and the orientation (longest principal axis) of the segmented chess piece piece (Fig. 5(c)).
4) Compute the global displacement and orientation of the piece in manipulator coordinates (Fig. 5(d)).
5) Align the end effector with the shorter principal axis (Fig. 5(c)) and move above the piece center (Fig. 5(d)).

IV. EXPERIMENTAL EVALUATION

The use of the depth camera and the palm camera enables Gambit to play a natural, uninterrupted game of chess on essentially arbitrary chess boards and uninstrumented pieces.

In 2010, Gambit participated in the AAAI Small Scale Manipulation Challenge [2], where it played chess games against three other robots. The four participating chess robots were quite different in many respects, including basic mechanical architecture, system cost, and software architecture. In each of its three pairwise matches, Gambit scored higher than its competition. Gambit also had the highest cumulative score. The informal feedback on the subjective experience of playing a game with Gambit has been generally positive, with examples of people playing multiple games, games played successfully without the system developers being present, etc.

In the following, we analyze Gambit's performance on the chess board and the three different chess sets shown in Fig. 7.

Our experiments were structured in three parts. First, we analyzed problems encountered with perception, manipulation, or logic during several *complete games* of chess encompassing several hundred turns. Second, we set up *specific grasping experiments* to test the improvement due to visual servoing. Third, we tested the accuracy of *piece recognition* by setting up several mid-game boards and attempting to locate occupied squares and identify pieces.

A. Perception and Manipulation

In order to test the baseline sensing, manipulation, and game logic, we conducted two complete games with each of the three chess sets shown in Fig. 7. Games were played both with a human opponent, and with the robot playing both sides.

For baseline games, we used a standard chess opening setup with no handicaps. Games averaged 112.3 chess moves. For each game, we tracked how many *interventions* were requested by Gambit, and how many *failures* occurred (with no request for intervention). As certain moves are more complex than

others—castling and capturing require multiple manipulation steps—we actually report on interventions and errors across all *manipulations* rather than across all moves. Six games provided 786 total manipulations. We distinguished between the following errors:

- Manipulation: Failing to grasp a piece, dropping a piece, or failure to place a piece legally, see Fig. 8.
- Perception: Failing to find a piece on the board (e.g., due to occlusion) or failing to detect illegal piece placement, see Fig. 9.
- Miscellaneous: Gripper collision with pieces, collisions between pieces during placement, or failure to find an inverse kinematics solution for some motion, see Fig. 9.

Logic errors (failure to correctly identify an opponent's move, illegal moves) are not reported because none occurred.

Manipulations	Autonomous Successes	Interventions Requested	Failures
786	720	38	28
100%	**91.6%**	4.8%	**3.6%**

Fig. 8. Successes, requests for intervention, and failures of manipulation.

Perception		Miscellaneous	
Interventions Requested	Failures	Interventions Requested	Failures
9/786	12/786	3/786	7/786
1.1%	1.6%	<0.1%	0.9%

Fig. 9. Perceptual/miscellaneous failures and requests for intervention during 786 manipulations. The first row is total occurrences across all six games. The second row shows the corresponding percentages.

Generally, the results demonstrate the reliability and robustness of the Gambit system. Most errors encountered were manipulation errors (Fig. 8), often caused by either a failure to pick up an off-center piece or illegal placement of a piece that was picked up awkwardly. Visual servoing can reduce these errors. Moreover, the Lewis Chessmen are difficult to grip because of their shape and relatively large for the chess boards we used and, resulting in a higher percentage of errors. Improved centering of the gripper over the piece can be expected to help with these difficulties.

B. Visual Servoing

Because the total manipulation errors are still relatively low, playing the number of games necessary to demonstrate improvements from visual servoing would be significantly time-consuming due additional image-processing overhead. Thus, servoing was tested with a different experimental setup designed to concentrate on manipulation cases that are likely to fail without servoing, e.g., difficult pieces that are *deliberately* placed as badly as possible.

From each chess set, we placed pieces in the extreme corners of a board square—the worst possible placement that might still be considered legal during a game—with other pieces on the surrounding squares to serve as visual distractors. Gambit then attempted to pick up each piece and put it down again several times, both with and without servoing. After each grasp, experimenters replaced the piece in the original position.

Regardless of how a piece was picked up, it was always put down by moving the gripper to the center of a chess square. Thus, an off-center grasp resulted in an off-center placement. Accordingly, experimenters also assigned a score describing how well-centered a piece was on the square after placement, as determined by the quality of the grasp. The five-point scale ranges from 1 (very poor grasp) to 4 (perfect grasp), with 0 assigned for illegal placement or a trial in which grasping failed. "Down-the-line" errors, where a poor but legal piece placement causes later grasping errors, are not captured by our experimental protocol. The results are summarized in Fig. 10.

	Num. Trials	Successful Grasps		Grasp Quality
Servoing	40	31	**77.5%**	3.1
No Servoing	40	7	**17.5%**	1.4

Fig. 10. Grasping quality of deliberately poorly-placed pieces with and without servoing. We report whether the grasp was successful and how well-centered the piece is in the gripper. Visual servoing provided a substantial improvement in success and grasp quality.

These results suggest that visual servoing can substantially reduce the number of grasping and placement failures encountered while playing games, particularly since this experiment was designed to reflect a worst-case scenario. However, we found that visual servoing, particularly piece detection, is somewhat slowing down the game play and sensitive to the frequently changing lighting conditions, leading to an additional source of possible failures.

C. Piece Recognition

We tested piece recognition on the Lewis Chessmen, which are less stylized and lower contrast than the other chess sets. We collected supervised training data from games. Identification models were trained according to Sec. III-B. We then set up two chess games in different states of completion, in order to test how accurately the system could recognize the pieces that are currently in use in a game: First, the depth camera was used to determine which chess squares were occupied, see Sec. III-A. Second, the robot hand moved above occupied squares to take pictures of each piece from directly above. Third, these images were used as test images of the piece recognition procedure detailed in Sec. III-B.

The confusion matrix for our experiments is shown in Fig. 11. On a test set of 59 images, four piece types were misclassified, which led to an overall accuracy of 93.22%.

Although the data set is small, the overall results suggest that close-range object recognition using the camera in the gripper has real promise in manipulation and interaction tasks.

V. FUTURE WORK

In the chess domain, reliable piece recognition can provide a principled way of resolving errors, such as perception uncertainties, e.g., occlusions: The gripper camera could simply be sent to look at areas of the board where problems were encountered. More generally, a reasonably high-resolution,

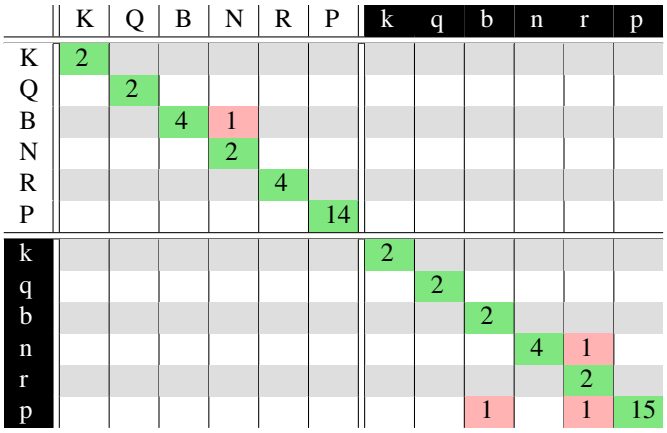


Fig. 11. Confusion matrix from recognition of 59 pieces participating in two chess games. Pieces are labeled using standard FEN notation, e.g., "N" is a white knight. The *color* of the chess piece (white/black) was 100% accurately determined, the recognition accuracy of the piece *types* was about 93%.

close-up image of the workspace can lead to improved grasp selection, better motion planning, and ambiguity resolution. Outside the chess domain, machine learning can offer additional intelligence in noisy, unfamiliar "real-world" tasks.

Our work can be generalized toward the ultimate goal of an intelligent manipulation assistant. Playing a wider variety of games, or even to learning completely new games, is one interesting direction for future work. Another one is to learn cooperative tasks that are less structured, e.g., not turn based. For example, future systems could assist with Lego or other building activities. Developing the ability to manipulate more challenging objects, such as playing cards, Monopoly money, or small game pieces would enlarge the set of cooperative tasks in which a system such as Gambit can participate.

Comprehensive perception of the human, e.g., monitoring the human's facial and hand gestures, would enable improved human-robot cooperation. This could allow for more natural interactions, such as taking back moves, or smoothly interleaving other activities with game play. These research questions have to be addressed to move beyond chess playing to realize the vision of an intelligent manipulation assistant.

VI. CONCLUSIONS

Robotic game playing is an excellent toy problem for exploring human-robot collaboration because it has an easily varied degree of structure. This paper presented Gambit, a robotic system that is capable of playing chess with a human opponent in a natural fashion, using a variety of ordinary, uninstrumented chess sets. For interaction with the human opponent, Gambit solely uses speech synthesis, its own body gestures, and perception of the board and the player's hands. It does not require any computer display, input devices, or buttons to play chess with a human opponent. Turn taking is completely natural, and does not require a "chess timer" to mark the end of a turn. Compared to prior chess-playing robots, Gambit operates with less structure: It does not use an instrumented chess board and can learn to play with a diverse variety of novel human chess boards/pieces. Gambit unambiguously beat all of its robotic opponents at the 2010 AAAI Small Scale Manipulation Challenge. Videos and more details on the Gambit project can be found at [1].

Although chess playing is not the end goal of this research, the fact that people have been trying to build chess playing automata for hundreds of years suggests that chess might be a successful application for a cost-effective small scale manipulation system. Since Gambit's design is open source, we hope that the community will iterate on the design to reduce its cost even further, and make small scale manipulation an everyday reality, for chess playing and many other human-robot collaboration applications.

REFERENCES

[1] http://www.cs.washington.edu/robotics/projects/gambit
[2] M. D. Anderson, S. Chernova, Z. Dodds, A. L. Thomaz, and D. S. Touretzky, "Report on the AAAI 2010 Robot Exhibits," *AAAI Magazine, in press*, 2010.
[3] L. Bo, X. Ren, and D. Fox, "Kernel descriptors," in *NIPS*, 2010.
[4] L. Bo and C. Sminchisescu, "Efficient Match Kernel between Sets of Features for Visual Recognition," in *NIPS*, 2009.
[5] A. Bratersky, "Dvorkovich, Chess Robot Go 1-1," http://www.themoscowtimes.com/news/article/dvorkovich-chess-robot-go-1-1/408089.html, June 2010.
[6] C. Breazeal, A. Brooks, D. Chilongo, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, and A. Lockerd, "Working Collaboratively with Humanoid Robots," in *Humanoids*, 2004.
[7] C. Breazeal and B. Scassellati, "Infant-like Social Interactions between a Robot and a Human Caregiver," *Adaptive Behavior*, pp. 49–74, 2000.
[8] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *CVPR*, 2005.
[9] K. Dautenhahn and A. Billard, "Games Children with Autism can Play with Robota, A Humanoid Robotic Doll," in *Cambridge Workshop on Universal Access and Assistive Technology*. Springer, 2002, pp. 179–190.
[10] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "LIBLINEAR: A Library for Large Linear Classification," *JMLR*, pp. 1871–1874, 2008.
[11] J. T. Feddema and O. Mitchell, "Vision-guided Servoing with Feature-based Trajectory Generation for Robots," *IEEE Trans. on Robotics and Automation*, pp. 691–700, 1989.
[12] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. of the ACM*, pp. 381–395, 1981.
[13] H. Kose-Bagci, K. Dautenhahn, and C. L. Nehaniv, "Emergent Dynamics of Turn-taking Interaction in Drumming Games with a Humanoid Robot," in *RO-MAN*, 2008.
[14] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *IJCV*, vol. 60, pp. 91–110, 2004.
[15] B. Mayton, L. LeGrand, and J. R. Smith, "An Electric Field Pretouch System for Grasping and Co-Manipulation," in *ICRA*. IEEE, 2010.
[16] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibsz, E. Berger, R. Wheeler, and A. Ng, "ROS: An Open-source Robot Operating System," in *Open-source Software Workshop of ICRA*, 2009.
[17] S. Schaffer, "Enlightened Automata," in *The Sciences in Enlightened Europe*. University of Chicago Press, 1999.
[18] B. Schölkopf and A. J. Smola, *Learning with Kernels—Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.
[19] T. Standage, *The Turk: The Life and Times of the Famous Eighteenth-Century Chess-playing Machine*. Walker, 2002.