A CHESS PLAYING ROBOT: LAB COURSE IN ROBOT SENSOR INTEGRATION

F.C.A. Groen, G.A. den Boer, A. van Inge, and R. Stam

Faculty of Mathematics and Computer Science
University of Amsterdam
Kruislaan 409
1098 SJ Amsterdam

Abstract: To obtain practical experience in intelligent robotics a lab course has been developed, in which students have to create a chess playing robot. The system has to detect the move of a human opponent and has to take the appropriate action. Beside the integration of sensing and control of the robot arm the students gain experience in working in a team to realize a medium size project. In particular the programming environment proved to be an important factor in an efficient course.

Introduction

Lab courses for students should be challenging and if possible should have a playful element. In the described robotics lab course, students have to create a chess-playing robot, able to detect the move of a human opponent, and to respond correctly with a next move. The students have to develop the sensing to detect the moves, the path planning and the inverse kinematics to control the robot arm and the integration of all the modules to an operational system.

The main objective of this lab course is to give the students in a curriculum of Computer Science the opportunity to gain experience with the problems of a real sensor-based robotsystem. The system has to operate in an unstable and changing environment (the chess board and the pieces may move), so pre-planned actions are not possible. The students have to find robust solutions to these problems and gain experience in off-line robot programming. In parallel to this course a lecture course on robotics and sensing is taught, which has to be applied in this lab course. This lecture course follows partly the book of Fu, Gonzalez, and Lee [1].

A second objective for the students is to gain experience in working together in a team on a medium sized project. In this project 6 students work together on 3 tasks (2 students on each task). The datastructures for communication between the tasks are given, but the integration of the 3 separate tasks together with supplied modules forms an essential element to end up with a fully operational system.

The lab course has to be completed within 10 weeks, 8 hours per week. During the period of this lab course also the robotics lecture course is followed. The degree of difficulty of the lab course requires a programming environment in which it is simply to experiment. It should also be easy to add routines, written by the students, without the burden of writing user interaction for the feasibility studies, hence prohibiting laborious programming. In the next section first the programming environment will be described. Thereafter an overview of the complete system will be given and the student tasks will be specified. At last some results and conclusions are given.

Programming environment

As programming environment the SCIL-IMAGE [2][3] package is used, which runs under UNIX and XWindows. SCIL is a multi-level environment originally developed in image processing, for feasibility studies. IMAGE is a image processing package linked to SCIL.

The lowest level is formed by a C-interpreter. Using this interperter complete C-programs can be developed. Interpreted code can be directly combined with pre-compiled code from incorporated libraries. If required these programs can also be compiled and linked to SCIL.

On top of the interpreter a command expander is placed. By supplying to SCIL a command description, (Command Description File) commands are automatically expanded to C-functions; lacking parameters are filled in with their default values and parameters are checked for their range. By extending the command description file the user can add his own command descriptions.

On top of the command expander a menu and dialogue generator is placed, using the same information from the command description file. As a result, apart from the C-code, functions can be called by direct commands or by menu-choices.

This environment showed to be an ideal basis for the robotics course. The supplied modules, such as the graphic simulator and the chess-module, are linked with the SCIL-IMAGE package. The C-code developed by the students can be easily loaded in the interpreter and executed. Error messages from the interpreter resulting from the program to be developed, give a clear view

0-7803-0640-6/92 $3.00 © 1992 IEEE

where an error occurred and how to solve it. In this way a fast feedback is obtained to evaluate the implementation of the functional design. When the C-code fulfils the requirements it can be linked with the other parts of the SCIL package, and executed at full speed without interpretation. In particular also the fact that the SCIL package has and automatic menu and dialogue-box generation makes it easy to realize feasibility studies, as the user interaction is already taken care off and it is not necessary to be programmed by the students. Particularly this environment makes feasibility studies easy and enables simple integration of image processing and robot control.

IMAGE is a powerful image processing package. It includes linear and non-linear pre-processing for image enhancement, and noise suppresion, image segmentation methods, mathematical morfological operations and image and object measurement techniques.

Overview of the lab course

The lab course is built around a six-degree of freedom UMI RTX robot arm [4], an ELTEC image acquisition system with an overhead Sony CCD camera, and the SCIL-IMAGE development environment running on a network of 12 SUN colour workstations. In figure 1 a photograph of the hardware set-up is given.

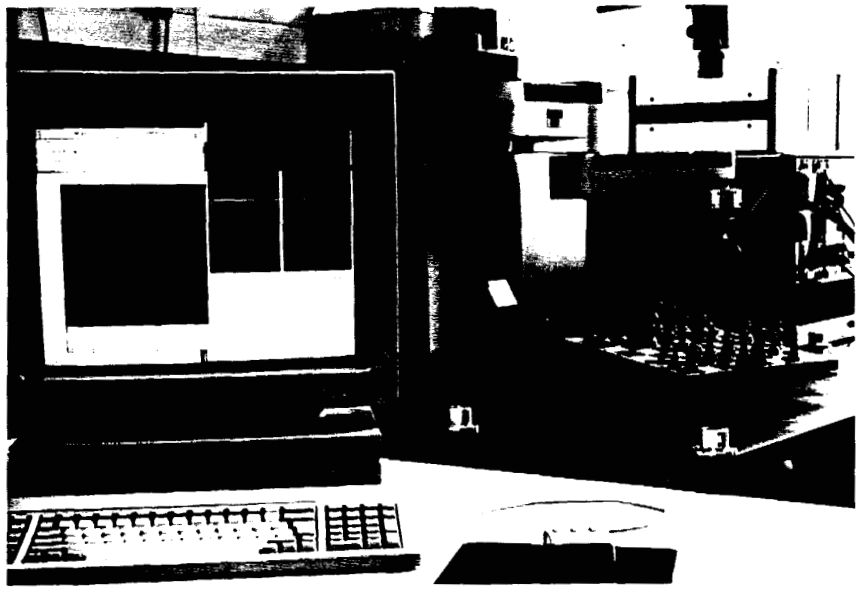

Fig. 1. The hardware set-up.

A diagram of the final configuration of the software system is given in figure 2. This figure shows the different modules and their interconnections. The modules labeled with task 1, 2, and 3 are the tasks the students have to realize. The other modules are supplied with the lab course.

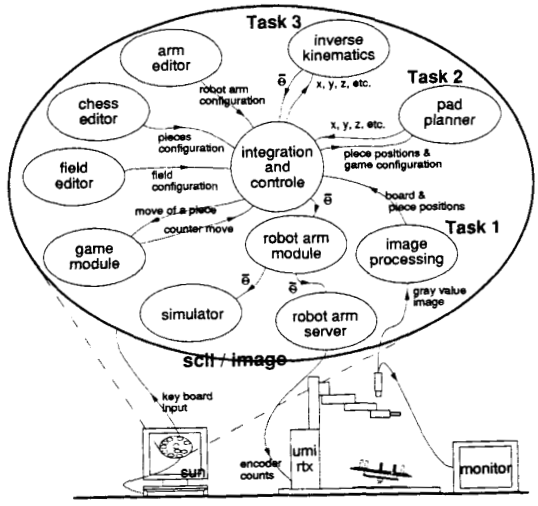

Fig. 2. Diagram of the software system configuration.

With the *image processing* module (task 1) the position of the chess-board and the moves of the pieces have to be detected. The symbolic notation of the move is fed into the chess *game module* (GNU chess [5]), which delivers the counter move. For this counter move a collision free path has to be planned by the *path planner* (task 2). This path is converted to joint coordinates by the *inverse kinematics* module (task 3).

The robot arm is actuated through the *robot arm* module. This module is crucial for off-line programming. It controls the *graphics simulator*, which shows the robot arm configuration on the workstation

screen. If the supplied joint values are out of bounds, the simulator will refuse to send the values to the real robot arm through the *robot arm server*. This prevents the robot from demolishing itself or its environment. When a program executes correctly in simulation (which can be performed on each of the 12 workstations), the program may be executed with the real robot.

The last task is formed by the integration of all the modules. Here the students combine there modules together with the supplied modules.

Supplied are also the editors for the different datastructures, describing the robot parameters, the initial position of the chess pieces on the board, and the position and orientation of the chess board. These editors facilitate the checking and debugging of the C-code developed by the students.

Task 1: Image processing module. Using the overhead camera and the image processing tools available in the SCIL-IMAGE package, changes in the grabbed images have to be detected. These changes can be moves of the pieces or changes in the orientation of the board. Because of the camera position only board movements parallel to the robot arm table surface are detectable. The tilt of the board is given by means of the board position editor and will be constant during the game. Development of a robust sensing strategy is essential for this step. In the first phase the students get acquainted with the image processing tools available and the influence of illumination. Thereafter they have to detect the moves and to transform them into a symbolic notation (for instance A2-B3). This detected move is sent to the chess game module which produces a counter move. In the final phase the solution has to be made robust. Board moves have to be detected and the datastructure is updated with the new board coordinates.

Task 2: Path planner module. First, the students experiment with cyclic reordering pieces on the board given in symbolic notation. Next the students have to produce a collision free path for the robot arm in order to execute the move of the chess game module. The students have to take into account the following three requirements. The different heights of the pieces, the size of the opening of the gripper, and the fact that the board can be moved during the game.

Next, methods for generation of collision free paths are exploited by not allowing paths above the other pieces, but instead to manoeuvre between the pieces on the board and to find the shortest path. See the illustrations in figure 3. A few distinct shortest path solutions can be made: one, a path around the board in case that there is no space between the pieces, two: a path which will go between the pieces and outside the board, and three, the path will go over the pieces. The last case is only allowed if there is no path found between the pieces. Among others this problem can be solved by wave-propagation, Dorst et al. [6]. This leads to the final goal of this task, solve an arbitrary positioning of the board in 3D. Therefore the students have to tilt the board a little and solve the problem again.

The path planner module produces sets of three dimensional carthesian coordinates and the orientation of the gripper. These sets are sent to an inverse kinematics module.

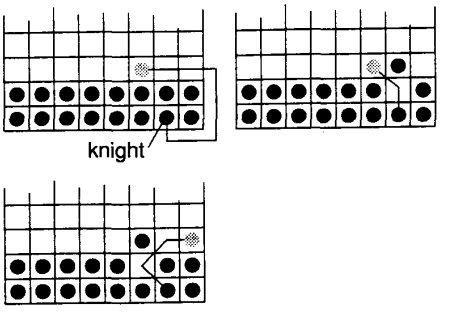

Fig. 3. Possible paths

Task 3: Inverse kinematics. This module accepts the sets of coordinates from the path planning module and produce sets of joint angles and gripper positions which are supplied to the robot arm, hence the inverse kinematics module must be programmed for three dimensional execution. The robot parameters are given in the Denavit Hartenberg notation. The inverse kinematics of the robot arm has to be solved and programmed using a geometric or algebraic approach. Here the students have to solve a problem originating from the difference between thereory and practice. Because of a hysteresis and an offset in the joints, there is a difference between the Denavit-Hartenberg model in the graphics module and the real robot. This difference shows up dramatically when the robotarm flaps its configuration (passes a singularity) and illustrated in figure 4. The students have to restrict the configuration of the robot or have to model this hysteresis and offset and compensate for it in the inverse kinematics module.

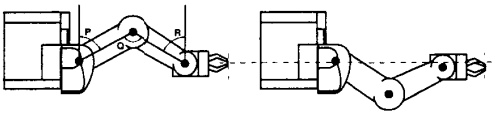

Fig. 4. Flapping of the arm

The final part is the integration of the modules to a complete working system. Interaction between the human and the computer interaction is at two points. First the human has to move a piece and second the human has to notify the computer of completion of the move.

Results and conclusions

Task 1: Most students solve the detection of moves of the chessboard and pieces by subtracting two images from each other. They first segment the images before subtraction and suppress noise disturbances by morfological operations (like openings and closings). Through object measurements the position of the pieces are found in image coordinates. The corners of the chess-board are found by subtracting an eroded image from the original thresholded version. Corners are found by the opening of this image. Hence from their position the position and orientation of the chess-board is calculated.

There were some interesting solutions to determinate the board position, without interference of the user. For this the students used a xy-coordinate system placed on the supporting table.

Other students tried to detect the completion of a human move. This to limit human-computer interaction. This failed because they could not differentiate with the illumination effects.

Task 2: Most interesting part here is collision avoidance with the other pieces on the board. Almost all groups solve this by wave propagation. In this case the 2D space is quantized and in the free space a pseudo-euclidean distance is calculated to the goal position. The students did not encounter specific problems, however, it took a lot of effort to produce the path planning module.

The students had difficulties in determining what belongs to path planning and what belongs to the inverse kinematics, hence they partly wrote code which had to do with to inverse kinematics.

Task 3:. In general the geometric approach is used in finding the solution to the inverse kinematics problem The students had difficulties in defining an experiment to determine the hystheresis and offset of the arm. They also had difficulties in executing the experiment.

In the opinion of the students the checking the robot work space was difficult.

Most remarkable was that in the beginning stage the students came twice so far using the SCIL environment, as compared to the course before without this environment. All students groups finished in the required time of 10 weeks. In the previous course this was 12 to 14 weeks and the results were at a lower level.

Using strict definitions of the datastructures, through which the modules communicate has the advantage that modules of different groups can be intermixed, which makes comparison easy, and gives the possibility to supply a module to a group of students in which for some reason one task was not completed, so that the integration can still be realized by the remaining students.

Acknowledgement

The authors wish to thank prof. dr. Smeulders for stimulating ideas and discussions.

References

[1] K. S. Fu, R.C. Gonzalez, C.S.G. Lee, Robotics, Control, Sensing, Vision and Intelligence, McGraw-Hill, New York, 1987.

[2] T. Ten Kate, R. Van Balen, A. W. M. Smeulders, F. C. A. Groen, G. A. Den Boer, "SCILAIM: A multi-level interactive image processing environment", Pat. Rec. Let., pp. 429-441, 1990.

[3] SCIL-IMAGE manual
University of Amsterdam for the centre of Image Processing and Pattern Recognition(CBP)
July 1, 1991

[4] UMI-RTX manual.
Published by Universal Machine Intelligence Limited, London, England
April 1987

[5] GNU chess
Free Software Foundation, Inc.
John Stanback

[6] L. Dorst, I. Manddhyan, K. Trovato, "The geometrical represntation of path planning problems", Robotics and Autonomous systems, pp. 181-196, 1991.