Collective Construction Using Lego Robots

Crystal Schuil[1], Matthew Valente[1], Justin Werfel[2], Radhika Nagpal[1]

[1]Harvard University, 33 Oxford Street, Cambridge, MA 02138
[2]Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139
schuil@fas.harvard.edu, valente2@fas.harvard.edu, jkwerfel@csail.mit.edu, rad@eecs.harvard.edu

Robot Exhibition: Extended Abstract

Social insects, such as ants and termites, collectively build large and complex structures, with many individuals following simple rules and no centralized control or planning [Theraulaz and Bonabeau 1995, Camazine et al. 2002]. Such swarm systems have many desirable properties: a high level of parallelism, cheap and expendable individuals, and robustness to loss, addition, and errors of individual insects.

Our goal is to design systems for automating construction that are similarly adaptive and robust, but build what we want. Automated construction will impact our ability to operate in inhospitable habitats, from outer space to under water, and allow automated disassembly and repair.

Recently, our group has developed a family of decentralized algorithms by which identically-programmed autonomous agents can collectively and reliably build user-specified solid 2D configurations from building blocks [Werfel et al. 2006]. In that system, the agents act without explicit communication, instead relying on the partially built structure as a form of indirect coordination similar to *stigmergy* in insects. The *extended stigmergy* approach is to enhance the environment (in this case, the building blocks) to store and process information. We showed that adding even simple capabilities to blocks, e.g. writeable state using RFID tags, can significantly increase the robustness, speed and simplicity of the system as a whole. The building process is robust to variable numbers of agents, asynchronous agent timing, and addition/loss of agents. The algorithms and analysis are available in [Werfel et al. 2006].

An important concern with the design of algorithms for mobile robots is their feasibility when implementing them in the real world. Our algorithms rely on simple robot behaviors, such as following the perimeter of building blocks and carrying and placing self-aligning blocks. The system does not rely on wireless communication, globally available position information, or perfect manipulation/ movement, which are challenging to implement in mobile robot systems. The algorithms are simple enough to be implemented on current low-cost robot hardware. In order to demonstrate the feasibility of the system, we have developed two hardware prototypes: (1) an Evolution Robotics ER1 based mobile robot (presented in [Werfel et al. 2006]) and (2) LEGO Mindstorms based robots.

In the AAI06 presentation we demonstrated the LEGO Mindstorms based prototype. Our current system has two robots, which autonomously construct 2D structures from flat building blocks.

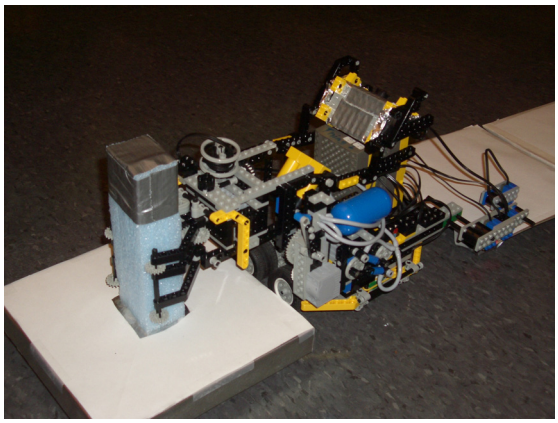

Figure 1: The LEGO constructor robot.

The LEGO Mindstorms robot implements the basic behaviors of the algorithm using its limited set of sensors. It can (a) locate the station for new blocks (b) find the existing structure (c) navigate the perimeter of a partially constructed structure and (d) recognize right and left corners in the perimeter. These behaviors are sufficient to implement an algorithm whereby robots always construct a contiguous structure free of gaps and tunnels. In order to implement these behaviors, the robot relies on two light sensors mounted in the front that can distinguish light from dark. The blocks are white and contrast with the black field. The robots are programmed using NQC, a C-syntax derivative specifically created for the LEGO Mindstorms platform. The basic algorithm is as follows: the robot starts at the perimeter of the field and searches until it finds the station for new blocks. It grabs a block and then heads towards the structure. It follows the perimeter of the structure until it finds an appropriate location to place the block. After placing the block it heads back to the

perimeter of the working area and repeats the process. The robot uses feedback from rotation sensors and light sensors to correct errors in movement, and can robustly follow a complicated perimeter.

Building our robots out of Lego offers many advantages compared to preconfigured robotics kits or machined parts. The robots are infinitely reconfigurable and lend easily to modifications and updates to the design. The availability and cheap cost of the Lego Mindstorms kits also enable our robot to be easily replicated, without having to use a machine shop or complicated equipment. From boxes of new parts, a fully functioning second robot was built in four hours.

The limited sensor/motor interface of the LEGO Mindstorms control unit (RCX), made it necessary to use two units per robot. The two RCX units control different aspects of the robot behavior and communicate with each other through infrared transmissions relayed off of a reflective surface. Each RCX unit contains three sensor input and three motor output ports. The primary RCX is connected to three standard Lego grayscale light sensors and two drive motors. The secondary RCX unit is connected to the pneumatic pump motor, the arm control servo switch motor, and the motor that opens and closes the grasping mechanism. The secondary RCX is also connected to two rotation sensors.

The primary RCX unit is dedicated to controlling the movement of the robot, including physical control of movement in addition to control and execution of the main algorithm. The primary RCX uses a control structure involving the simple tasks of line following, searching for the structure and picking up and dropping the block to implement the majority of the physical actions involved in the execution of the algorithm. The secondary RCX unit is responsible for running the pneumatic pump, moving the arm, and taking readings from the rotation sensors. It provides most of the ancillary functions of the robot, including the entire action of picking up and dropping blocks. It also provides feedback to the primary RCX through the rotation sensors

The rotation sensors are connected to the motors through the gearing of the wheels. The gearing gives the rotation sensors an approximate wheel rotation to rotation sensor rotation ratio of 4.8:1. The rotation sensors count each sixteenth of a turn of the input shaft. The built in accuracy combined with the gearing ratio enough accuracy and precision to be used for dead reckoning. By counting the number of rotations, we are able to achieve fairly precise specific movement, allowing many of our functions to utilize absolute distances in their execution. This also holds for turns. By adding and subtracting the values from the left and right side, it is possible to measure with high presicion the angle which the RCX has turned. Overall, the rotation sensors provide invaluable feedback that enable the robot to function without dependence on external direction mechanisms.

The RCX units also have built in timers that can be used to time the duration a motor has been running. This approach can also be used to turn the robot a set amount or a set distance The problem with this approach, a problem familiar to all roboticists, is that of battery power. The motors have rotation speed and peak torque that varies greatly with the level of battery power. Constant recalibration is necessary in order to achieve predictable behavior when using the timing mechanism rather than the dead reckoning mechanism employed with the rotation sensors. For our purposes, the continual recalibration is unacceptable and therefore we have attempted to avoid relying on this functionality for all but the most basic and non-critical functions. However, using the rotation sensors requires two sensor ports, forcing us to offload the motor feedback to the secondary RCX, which does not control the motors.

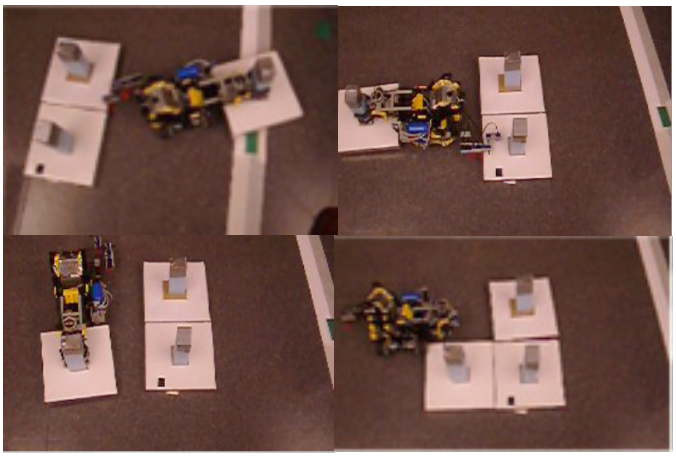

Figure 2: A sequence of stills from a video of the construction process.

Ideally the primary RCX unit would be wholly responsible for movement and executing the algorithm, while the secondary RCX would be used to operate the pneumatic arm. Unfortunately, the limitation of only three sensor ports per RCX makes this configuration impossible. A result of this, as previously mentioned, is the offloading of the rotation sensors to the secondary RCX. The effect of this is that whenever the primary RCX needs to make a 90 degree turn or move forward an absolute distance, it must send a message to the secondary RCX requesting that it reply with a “stop” message when a specified number of rotations has been reached.. The messages sent through infrared by the RCX’s consist of an 8-bit integer, from 0-255. For each absolute distance or turn function, a different message is sent, and the secondary RCX acts appropriately, switching on the specific message integer received. After the predefined absolute distance has been reached, the secondary RCX will send a generic stop message. Initially it was thought that the RCX receiving a message would need to send back a message confirming it received the first message. However, after substantial testing it was determined that the message transmission failure rate was

exceedingly low (none observed through multiple trials), and that the message could be presumed to be both transmitted and received.

The robot has a manipulating arm with which it can grab, raise and lower blocks. This arm is implemented using pneumatic cylinders and a worm-gear driven grabbing mechanism. The grabbing mechanism uses a chain-driven worm gear setup to close a hand with high enough force to hold a block suspended. The chain drive utilizes a clutched gear to avoid stalling the motor. In practice, the force of the grip is limited by the maximum torque of the clutch gear, approximately 1.5Nm. The lifting mechanism works through a pneumatic pump providing the pressurization of a series of pneumatic cylinders. The pneumatic pump utilizes two cylinders in a dual-acting design to ensure constant pressure. It runs continuously as soon as the secondary RCX's program is started. The pump, with the help of an airtank, is able to build sufficient pressure and volume to successfully pick up a block and hold it. A motor with a clutched gear serves as a servo and controls a valve which governs the air-flow to the bank of cylinders. When a block is detected, the grabbing mechanism rotates, gripping the block firmly. The servo is then activated and the fully pressurized pneumatic cylinders raise the arm. In order to place the block, the robot first releases the block and then lowers the arm.

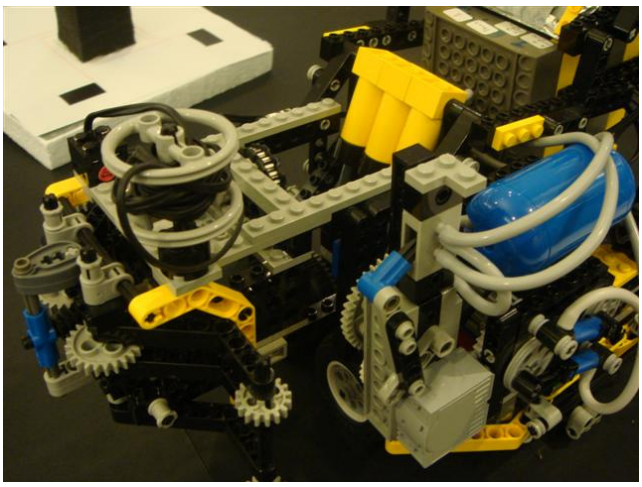

Figure 3: Close up of robotic arm.

The blocks that the robot uses to build the structure are custom-made from low-density foam to save weight. The base is cut from Styrofoam stock, while the handle uses polyurethane. Though our robot is able to turn through specified angles accurately, depending on the starting orientation of the robot, it may end up at an incorrect orientation and may not place the block at a right angle to the existing structure. In order to ameliorate this problem, we have made the blocks self aligning. Each of the building blocks contains eight neodymium magnets placed two to a side on the block. The attractive force of which causes a block dropped in a small tolerance to 'snap' to the existing structure in the appropriate orientation. This automatically corrects for small errors in block placement and frees the robot from having to align blocks with high precision. Although it does not make a difference to the overall algorithm, releasing the grip on the block before lowering the arm is an important consideration. The attractive force between the block the robot is carrying and the existing structure can cause the robot to tip over if the block is not allowed to freely move into position at close distances.

The design of the robot is completely modular. No glue was used in the construction of the robot. All parts can be removed and replaced. The gripping arm, air pump, servo, and sensor arm are completely structurally independent of each other. This enables us to easily switch out and modify parts without having to modify the rest of the robot. If different blocks are used, for example, it is a simple task to detach the gripper arm from the base and adjust it to a new size requirement.. Another advantage of the modular construction is that it is relatively easy to replace a broken or worn down part. Instead of having to manufacture a new part or ordering a costly replacement, LEGO parts are completely interchangeable and a new part can be switched in for an old one. However, the modular construction means that through natural vibrations, various pieces can occasionally fall off of their own accord, increasing the need for preventative maintenance and occasional repairs. Despite this, our design has evolved to a mechanically sound enough state that the only maintenance it needed during five straight hours of operation at AAAI06 was a battery change.

The light sensors are simple phototransistor grayscale sensors coupled with LED's. The light sensors can return a value from 1-100 based on the amount of light reflected from a surface. Although this is the theoretically possible light range, in practice even over a semi-reflective black surface the lowest value is 30 and under 60 for bright white reflective surface. The robot adapts to slight changes in lighting conditions through calibrating the values of light and dark at the start of each run of the program. The brightest value from the top of the block is measured and the darkest value of the floor is measured. Predefined thresholds are applied to these values to ensure tolerances in slightly varying light levels. The threshold values have been determined through testing in many different lighting conditions. The light sensors are very sensitive to changing light conditions and floor conditions. It is necessary to calibrate the sensors at the beginning of every run because even moderate changes in light levels during operation, for example, the setting of the run in a room which relies on natural light, can adversely affect the performance of the robot. If ambient light levels change significantly, it is necessary to adjust the predefined tolerances to ensure proper operation. With very glossy floors or very poor ambient lighting, the robot becomes extremely unreliable. In order to eliminate problems caused by reflective floors, a transportable demonstration area was constructed using heavy black poster board and white tape. This also allows

us to easily change the size and shape of the demonstration area. With careful calibration, some shades of gray can actually be detected, but this requires much smaller thresholds, and therefore the likelihood of incorrect identification increases. In order to minimize light sensor errors, the blocks and flooring used are only black and white. This allows the robot to only have to differentiate between light and dark instead of additional middle states. In order to overcome the limitations presented by an effectively binary grayscale sensor, we are currently testing the first prototype of a home-built color sensor. A color sensor will allow our robot to identify specific blocks and thus use the algorithm to respond to a map and build predefined structures.

The programming language that comes with the Lego Mindstorms system is called RCX. RCX Code uses a graphical interface with pictures and blocks in order to make programming easy for those with no programming experience, the target audience of the Mindstorms kit. We use the NQC programming language, an application-specific C derivative. NQC allows a programmer to use C-syntax to control the robot at a much lower level. The programming environment, Bricx Command Center, is available free online for downloading and has a comprehensive help library that lists all of the functions that can be used for the various types of RCX units in NQC.

The RCX units run multithreaded. In NQC, these individual threads are called "tasks". The more tasks that are being run simultaneously, the slower each task performs. The RCX does not allow prioritizing of any specific task. After much testing, we reached a careful balance of processing loads in each concurrent task. We manually prioritize each task by allowing certain tasks to end other tasks, reserving the processing power for a single action. After the interrupting task has completed, the program defaults to a control loop that specifies the point in the algorithm where the robot was interrupted, and starts the appropriate tasks again. An example of this is the task interrupting the line follower task. While the robot is driving around the perimeter of the demonstration area looking for a new block, two tasks are running simultaneously. One simply follows the perimeter of the area, using the light sensors to determine the edge between light and dark, and controlling the motors to move along this edge. The other task reads the third light sensor, and looks for a dark spot which indicates a block ready to be picked up. When the second task identifies a block, it stops the line following task, starts a task in charge of picking up the block, then stops itself. The line following task is not used again until the robot has reached the structure and starts to follow the perimeter of the structure in order to locate an appropriate place for placing the block.

Line following uses the two inner light sensors on the robot. The middle sensor reads the reflected light value, and depending on the value, the motors will turn slightly left or right, in essence snaking the robot along the edge of the block. If a light surface is detected, the robot is instructed to turn left. If dark, the robot turns right towards the structure. It is through continuous minute adjustments that the robot follows a perimeter. The innermost sensor should remain on the dark surface. If this sensor detects light, then the robot has either veered in towards the structure, or the robot has encountered a right angle in the structure and needs to turn in order to avoid running into it. In either case, corrective motion is to employed in order to avoid a collision

The third and outermost sensor is dedicated to finding dark locations when the robot is engaged in line following. We use dark demarcations on the surface of the block to indicate that a certain position with respect to a block has been reached. These demarcations are placed in towards the center of the block in such a spot that only the third sensor is able to identify them. When a dark spot is identified, the algorithm reacts to the presence of a block. If the robot is searching for a block to pick up and bring to the structure, it will move to pick up the block. If the robot is searching for a place to place the block and identifies a dark spot, the robot will move forward a set distance. The third light sensor again looks for a dark spot, which in this case indicates a possible appropriate location to place the block it is carrying. If the spot is dark, it indicates that the edge of the structure has been passed and the robot is now next to a position where a new block may be placed. Separate tasks monitor each of these possibilities.

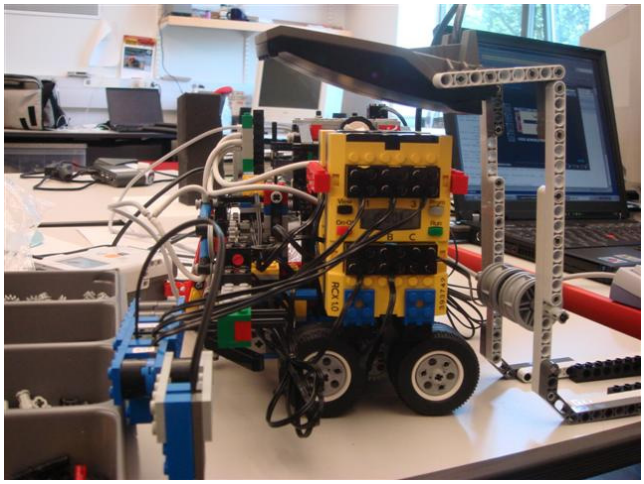

Figure 4: Close up of light sensor arm and the RCX units being reprogrammed.

Hysteresis is an important component in block identification and block placement position detection. If the robot immediately reacts to changes in light levels, it can trigger erroneously due to cracks between the blocks, or in shadows in the demonstration area. When the robot sees a black spot identifying a block, it utilizes a timer to ensure that the dark spot is constant for a specified amount of time. After trial and error, the appropriate amount of hysteresis was determined to be on the order of

milliseconds. This prevents identification from cracks between blocks and many other false positives, but does not stop identification if the robot quickly passes over a demarcation while turning or many other situations where the robot might miss a block.

Once the robot has a block and is looking for an appropriate site for block placement, a task runs that uses the rotation sensors to determine whether the robot has just turned a corner. If the robot identifies a block immediately after making a right corner, it will always place the block in the corner, filling it in. If the robot reaches the end of a column of blocks, it will place a block adjacent to the block at the end of the column. In this way, the robot will build an arbitrarily large sheet without creating holes or tunnels into which it cannot travel when placing additional blocks. The largest source of error in placing blocks is the initial location and orientation of the robot when the maneuvers to place the block begin. The robot does not always detect the black spot at the same location. The robot can be slightly further forward or back when the turning sequence begins as well as oriented slightly to the left or to the right. This occurs because of how the robot line follows: the robot is continually making small left and right adjustments and may detect the black spot while not completely parallel to the structure. The combination of these two factors results in slightly inconsistent block placement. However, these inconsistencies fell into two major categories: placement after the robot has just made a right turn, and placement after it had just completed a left turn. Since the secondary RCX controls the absolute distance/angle values of the placement maneuvers, the aforementioned corner recognition program also adjusts these values based on the whether a turn has recently been completed. This task employs timers, which gives a certain time frame for completion of the turn. This prevents the modified placement values from being used if the turn was completed over a certain time prior to identification of a valid placement position. Combined with the magnetic attraction of the blocks, this method is enough to counteract the innate accuracy problems with the Lego robots.

The LEGO robot system has many advantages: it is easy to extend the mechanical design to incorporate new capabilities, and it is easy to program using NQC. At the same time the limited sensors pose some difficulties. Also the limited interfacing ability makes it difficult to add new sensor capabilities, such as a camera or RFID reader. The next generation of the Lego robotics platform, appropriately named "Mindstorms NXT", has Bluetooth connectivity and will hopefully alleviate many of the problems the current implementation faces.

Our current systems form only 2D structures, yet insects form complex 3D structures that are well adapted to their environmental surroundings. In the future we plan to focus on building walled compounds and 3D structures, as well as building structures whose shape is partially determined by the environment. These systems pose many research challenges, in both algorithm design and implementation.

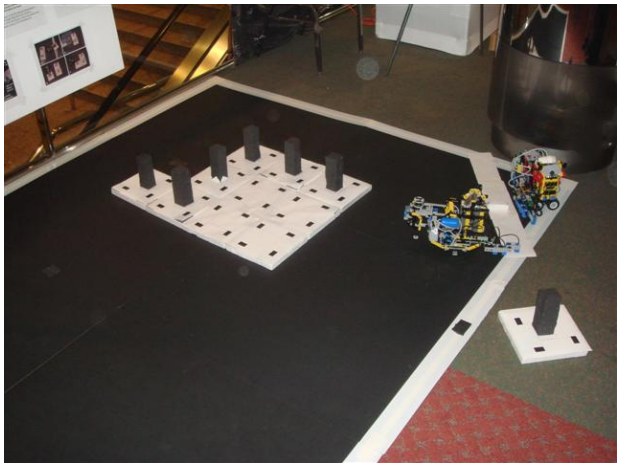

Figure 5: Robots at AAAI06.

We demonstrated this system at AAAI. The robots ran continuously for nearly two days. The robot body was very robust; however, we found some flaws in the placement and strength of the block attachment. In the future we plan to achieve more accurate block placement through looking back into the engineering of the block and into adjusting how the robot turns and places it.

This research is supported by the National Science Foundation (NSF) and a Clark Grant from Harvard University.

References

G. Theraulaz, E. Bonabeau, *Modeling the collective building of complex architectures in social insects with lattice swarms*, Journal of Theoretical Biology, 1995

Camazine, Deneubourg, Franks, Sneyd, Theraulaz, Bonabeau, *Self-organisation in Biological Systems*, Princeton University Press, 2002.

J. Werfel, Y. Bar-Yam, D. Rus, R. Nagpal , *Distributed Construction using Enhanced Building Blocks*, Proc. of IEEE International Conference on Robotics and Automation, May 2006.