MEAM 620 Final Project Report: Simulation of a Multi-Robot Puzzle Assembly Task

Spring Berman

May 9, 2008

1 Introduction

The objective of this project was to simulate a decentralized assembly task in which a group of robots assembles many identical puzzles from several different kinds of static pieces. The pieces are distributed randomly in the workspace and can bond to each other and to robots with magnets. In this scenario, the robots bond the pieces according to a specific assembly plan. This situation will eventually be compared, in terms of assembly yield and time to completion, to a scenario in which pieces collide randomly and decide probabilistically whether to stay bonded. The task was simulated using Webots software (http://www.cyberbotics.com/), which can readily model large groups of robots and magnet-outfitted pieces.

2 Simulation Setup

The assembly task was simulated in a walled hexagonal area using the Khepera III, which had a pre-existing model in Webots, as the robot platform. The Khepera III (Figure 1a) is a miniature mobile robot developed by K-Team SA in Switzerland (www.k-team.com). It is a differential-drive robot equipped with nine infra-red distance sensors, five ultrasonic sensors, and two infra-red ground proximity sensors. It was modeled with an emitter and receiver to enable radio communication. To pick up puzzle pieces, a bar with a rotational servo at the tip was attached to the robot (Figure 1b), and a magnet was placed on the exposed end of the servo. This magnet locks to a magnet on the top face of a piece (Figure 1c), and the servo is used to rotate the locked piece into the correct orientation for assembly. Pieces lock to other pieces via magnets on their side faces. Each piece was also equipped with an emitter and receiver for radio communication.

All emitters were given a range of 40 cm (the radius of the emission sphere) and a baud rate of 57600 bits per second. Two magnets can lock if the origins of their coordinate systems are within a distance of 2 cm and if the angle between their central axes is less than 0.2 radians.

The robots lock the pieces together according to the configurations shown in Figure 2. There are four different types of pieces (states 1–4), which can be combined in the configurations shown in states 5–8, where 8 is the final assembly. The assembly plan is: $1+2 \rightarrow 5$ and $3+4 \rightarrow 6$ in parallel, $5+6 \rightarrow 7$, $7+2 \rightarrow 8$. This sequence was chosen because the piece combinations are relatively easy to bond together at the specified orientations. Also, defining two steps that occur independently (in parallel) can speed up the assembly process.

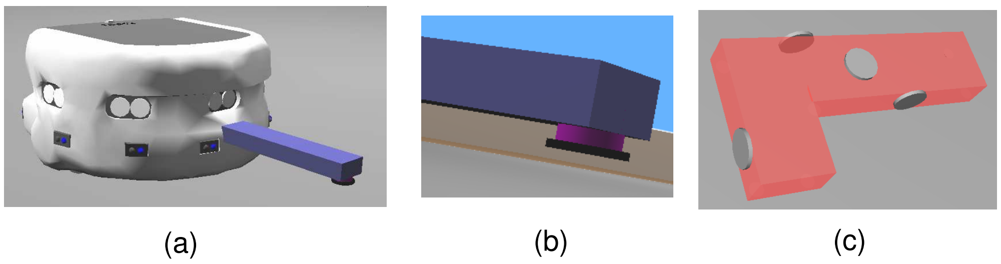

Figure 1: Simulation platform. (a) Khepera III robot with protruding bar; (b) close-up of bar; (c) puzzle piece.

3 Description of Robot and Piece Controllers

The simulation is comprised of a world file, a robot controller that is run by each robot, and a piece controller that is run by each piece. The world file defines the environment and the robot and piece models. The controllers are executable files (coded in C++) that define the actions of the object they govern. In this simulation, the robots and pieces switch between action states based on information they receive via *local* sensing and communication, which leads to a decentralized system that is scalable in terms of both vehicles and pieces.

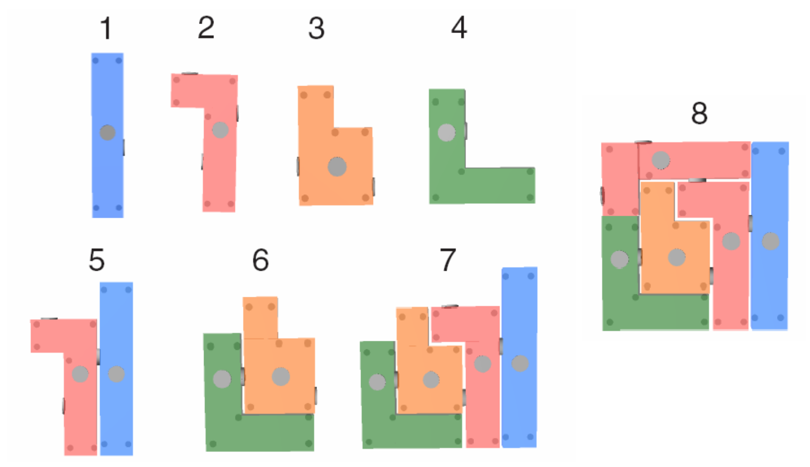

Figure 2: Piece types and allowable piece configurations in the assembly plan.

3.1 Robot Controller

Each robot is governed by a finite state machine whose states and associated transitions are described in the following sections. At each time step, defined as 32 msec, the robot updates its infra-red distance sensor values, the status of its magnet (locked or unlocked), and its left and right wheel speeds, $\dot{\phi}_1$ and $\dot{\phi}_2$. It also emits a message containing its unique ID, current state, and the type and unique ID of the piece it's carrying, which are both 0 if it isn't carrying a piece.

3.1.1 SEARCH_FOR_PIECE

The robot avoids collisions with the walls and other robots (Figure 3) using a Braitenberg controller, which computes the wheel speeds according to [1]:

$$\dot{\phi}_i = b_i + \sum_{j=1}^{N} \mathbf{W}_{ji}(1 - d_j/r) \ , \qquad i \in \{1, 2\} \ , \tag{1}$$

where b_i is a speed bias, N is the number of infra-red distance sensors, $\mathbf{W}$ is a matrix of weights, d_j is the value of the j^{th} distance sensor computed from a lookup table, and r is a range. The robot stops when it receives a message from a piece on the ground in state FREE, stores the piece's type and ID, and enters state ALIGN_WITH_PIECE.

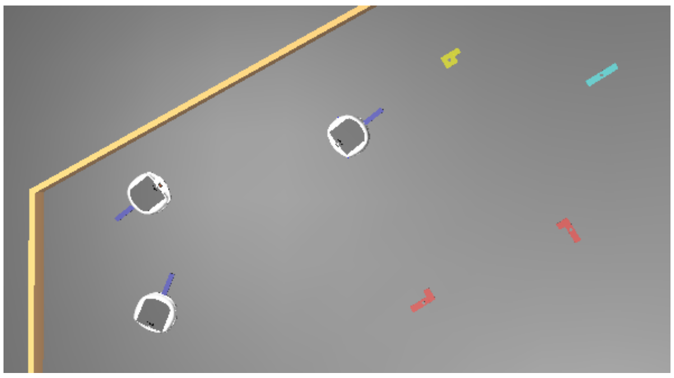

Figure 3: Robots searching for pieces while avoiding walls and other robots.

3.1.2 ALIGN_WITH_PIECE

Using the built-in function `receiver_get_emitter_direction()`, the robot obtains the normalized $[x\ y\ z]$ vector of the direction of the piece's emitter with respect to the robot's coordinate system. The robot's x and z axes are in the plane of the ground, with the z axis pointing directly behind it. The robot spins ($\dot{\phi}_1 = -\dot{\phi}_2$) until $|x| < 0.04$ and $|z + 1| < 0.03$, at which point its protruding bar, and thus its magnet, is approximately aligned with the piece's emitter, which has the same x, z coordinates as the magnet on its top face (Figure 4a,b). Then the robot enters state APPROACH_PIECE.

Error checks: If the robot remains in this state for more than 700 time steps, it backs up for 200 time steps and then reverts to state SEARCH_FOR_PIECE. If the robot receives no messages for more than 30 time steps, it reverts to SEARCH_FOR_PIECE.

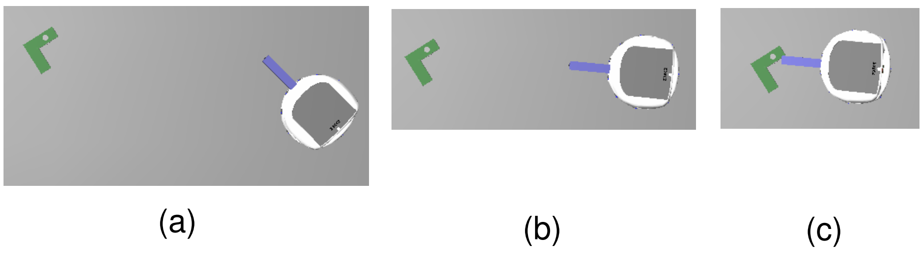

Figure 4: Robot (a), (b) aligning with a piece and (c) approaching it.

3.1.3 APPROACH_PIECE

The robot moves forward until the status of its magnet changes from unlocked to locked, which means that its magnet has locked to the piece magnet (Figure 4c). It then enters state ROTATE_PIECE.

Error check: If the robot remains in this state for more than 700 time steps, it backs up for 200 time steps and then reverts to state SEARCH_FOR_PIECE.

3.1.4 ROTATE_PIECE

The robot receives a message from the piece it is carrying that contains the angle the piece must rotate to be properly oriented. The robot turns its rotational servo by this angle (Figure 5), and once it receives a message that the piece is in state CARRIED, it enters state SEARCH_FOR_ROBOT.

Error check: If the robot remains in this state for more than 700 time steps, it unlocks its magnet (which releases the piece), backs up for 200 time steps, and then reverts to state SEARCH_FOR_PIECE.

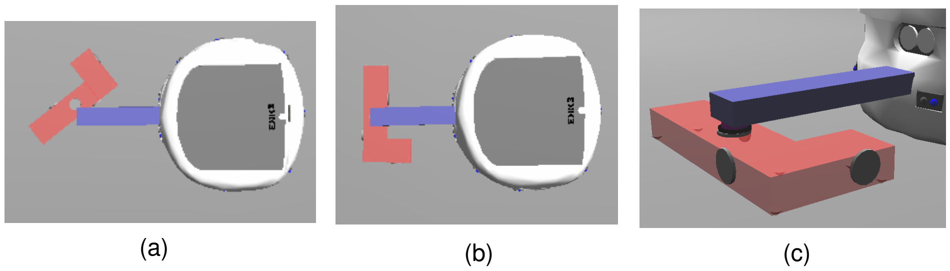

Figure 5: (a) Robot bonding to piece; (b) completed piece rotation; (c) close-up of bonded magnets.

3.1.5 SEARCH_FOR_ROBOT

The robot follows the Braitenberg controller and stops when it receives a message from a robot that is also in this state and is carrying a compatible piece, according to the assembly plan. For example, the robots in Figure 6a will stop to connect their pieces, while the robots in Figure 6b will ignore each other and keep moving. If the robot is carrying a type 2 piece (Figure 2) and it meets a robot carrying a type 7 piece, the robot will rotate its piece by π radians so that it fits into the assembly. If it has already rotated the 2 piece (but failed to lock it to a 7 piece) and encounters a type 1 piece, it will rotate it back to the original orientation. The robot next enters state ALIGN_WITH_ROBOT.

Error check: If the robot's magnet status is unlocked (the piece has come off somehow), the robot reverts to state SEARCH_FOR_PIECE.

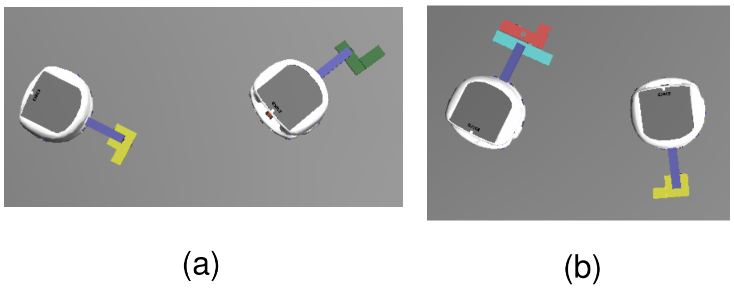

Figure 6: (a) Piece connection is in the assembly plan; (b) piece connection is not permitted.

3.1.6 ALIGN_WITH_ROBOT

The robot with the lower ID number becomes "robot 1," and the other becomes "robot 2." Robot 2 stops and enters state WAIT when it receives a message from robot 1 that it is in state ALIGN_WITH_ROBOT. Robot 1 aligns with robot 2's emitter in the same way that a robot aligns with a piece's emitter in state ALIGN_WITH_PIECE (Figure 7a,b), and then enters state WAIT. Once robot 2 re-enters this state from WAIT and receives a message from robot 1 that it is in state WAIT, it aligns with robot 1's emitter (Figure 7c,d) and then enters state APPROACH_ROBOT.

Error checks: If the robot remains in this state for more than 700 time steps, it backs up for 200 time steps and then reverts to state SEARCH_FOR_ROBOT. If the robot receives no messages for more than 30 time steps, it reverts to SEARCH_FOR_ROBOT.

3.1.7 WAIT

If the robot is identified as robot 2 and receives a message from robot 1 that it is in state WAIT, the robot enters state ALIGN_WITH_ROBOT. If the robot is identified as robot 1 and receives a message from robot 2 that it is in state APPROACH_ROBOT, the robot enters state APPROACH_ROBOT.

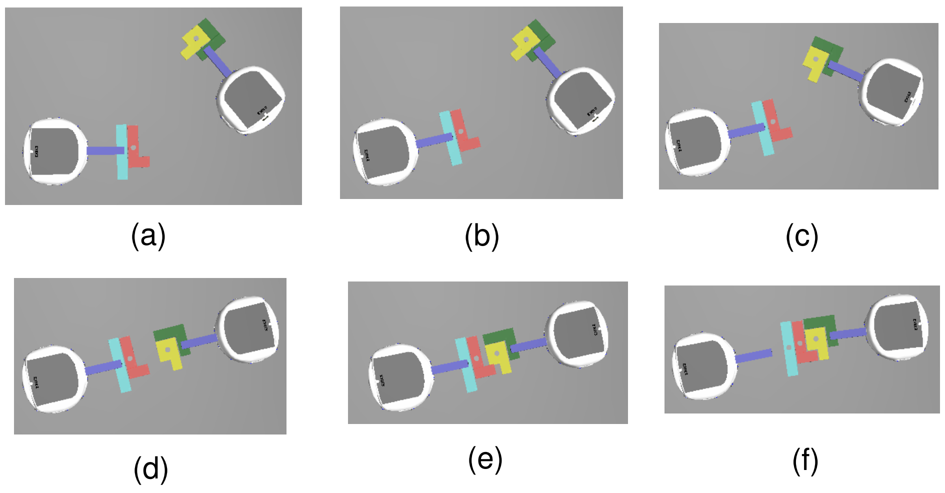
Figure 7: (a) Robots encounter each other; (b) first robot aligns; (c)-(d) second robot aligns; (e) robots approach each other; (f) the new piece is formed and one robot detaches.

Error checks: If the robot remains in this state for more than 700 time steps, it backs up for 200 time steps and then reverts to state SEARCH_FOR_ROBOT. If the robot receives no messages for more than 30 time steps, it reverts to SEARCH_FOR_ROBOT.

3.1.8 APPROACH_ROBOT

The robot moves forward (Figure 7e) until it receives a "locking" message from the piece it's carrying that has bonded to the other robot's piece. The robot stops and enters state SEPARATE_ROBOTS.

Error check: If the robot remains in this state for more than 700 time steps, it backs up for 350 time steps and then reverts to state SEARCH_FOR_ROBOT.

3.1.9 SEPARATE_ROBOTS

If the robot was carrying piece 2, 3, or 5, it unlocks its magnet and thus detaches from the newly formed piece (Figure 7f). To orient itself away from the other robot, the robot backs up for 25 time steps and then spins for 25 time steps. If the robot was carrying piece 1, 4, or 6, it stores the new type of piece that has been formed, which was in the locking message received in state APPROACH_ROBOT. If it was carrying piece 6 (and thus is now carrying piece 7), it rotates the new piece by $-\pi$ radians for proper assembly alignment. If the robot detached from the piece, it enters state SEARCH_FOR_PIECE. If it is carrying the new piece, it enters state SEARCH_FOR_ROBOT unless it was carrying piece 7, in which case it is now carrying the completed assembly. In this case, the robot detaches from the assembly and enters state SEARCH_FOR_PIECE.

3.2 Piece Controller

Each piece is governed by a finite state machine whose states and associated transitions are described in the following sections. At each time step, defined as 32 msec, the piece updates the status of its magnet (locked or unlocked).

3.2.1 FREE

The piece emits a message with its unique ID and current state. If it receives a message from a robot in state ALIGN_WITH_PIECE, APPROACH_PIECE, or ROTATE_PIECE that contains the piece's ID, the piece enters state CLAIMED.

3.2.2 CLAIMED

The piece emits a message with its unique ID and current state. If the status of the magnet on its top face changes to locked, which means that a robot has attached to it, the piece enters state ROTATING.

Error check: If the piece remains in this state for more than 200 time steps, it reverts to state FREE.

3.2.3 ROTATING

The piece's z axis points outward along the central axis of the magnet that connects to another piece. The piece obtains the normalized direction $[x\ y\ z]$ of the robot's emitter with respect to its coordinate system and emits a message with the angle it must turn for proper assembly, $\tan^{-1}(x/z)$. When $|z-1| < 0.03$, which means that the magnet that connects to another piece is approximately pointing in the direction of the robot's forward travel, the piece enters state CARRIED.

Error check: If the magnet on the piece's top face unlocks, the piece reverts to state FREE.

3.2.4 CARRIED

The piece emits a message containing its ID and state as long as it receives a message that the robot carrying it is still in state ROTATE_PIECE. When the piece's outward-facing magnet is locked, which means that the piece has bonded to another piece, the next state it enters depends on the following criteria. If the piece is type 1 or 4, it enters the NULL_STATE. If the piece is 3, it enters PIECE_6. If the piece is 2, the next state depends on the type of the other piece, which is communicated in a message from the robot carrying it: the piece enters PIECE_5 if the other piece is 1 and PIECE_8 if the other piece is 7.

3.2.5 PIECE_5

If the piece receives a message from the robot carrying it or the robot carrying the other piece that contains the piece type 1 or 2, meaning that the robot does not recognize the new piece type that was formed, the piece emits a message containing the word "Lock", the number 5, and the robot ID. If the piece is type 2 and its outward-facing magnet locks, the piece enters state PIECE_7.

3.2.6 PIECE_6

If the piece receives a message from the robot carrying it or the robot carrying the other piece that contains the piece type 3 or 4, meaning that the robot does not recognize the new piece type that was formed, the piece emits a message containing the word "Lock", the number 6, and the robot ID. If the piece is type 3 and its outward-facing magnet locks, the piece enters state PIECE_7.

3.2.7 PIECE_7

If the piece receives a message from the robot carrying it or the robot carrying the other piece that contains the piece type 5 or 6, meaning that the robot does not recognize the new piece type that was formed, the piece emits a message containing the word "Lock", the number 7, and the robot ID. If the piece is type 2 and its outward-facing magnet locks, the piece enters state PIECE_8.

3.2.8 PIECE_8

If the piece receives a message from the robot carrying it or the robot carrying the other piece that contains the piece type 2 or 7, meaning that the robot does not recognize the new piece type that was formed, the piece emits a message containing the word "Lock", the number 8, and the robot ID.

3.2.9 NULL_STATE

No actions; piece remains in this state indefinitely.

4 Simulations and Future Work

The assembly task was tested with 4 robots and 5 pieces that can be arranged to form the complete puzzle in Figure 2. The robots were able to successfully assemble the puzzle, although the pieces had to be assigned very low weights in order to prevent piece 7 from creating a large moment about the protruding bar on the robot that interfered with the robot's motion. To remedy this problem, in future implementations the robot will bond to piece 7 at a more centrally located magnet on the piece. In addition, the Khepera robot model will be replaced with a model of the Scarab robot in the GRASP laboratory in preparation for experimental implementation of a proof-of-concept assembly demonstration with two robots and two pieces. The simulation will also be scaled up to a large number of robots and pieces, and the mean and variance of the assembly yield will be measured and compared for varying robot and piece populations.

The assembly yield and time to completion for the scenario simulated in this project will eventually be compared to those of a stochastic self-assembly simulation. In this case, the pieces will be carried by robots and form magnetic bonds by colliding randomly. Upon collision, the pieces will probabilistically decide whether to remain bonded or detach. The probabilities will be designed to maximize convergence to a large number of complete assemblies while minimizing the number of "reactions" still occurring at equilibrium. It is expected that the simulation with planned bonding will outperform the stochastic simulation up to a certain degree of error in the "planned" simulation.

Acknowledgments Many thanks to Loïc Matthey for his advice and assistance in creating the world file and debugging the controllers.

5 References

[1] braitenberg.c. In the folder \projects\default\controllers\braitenberg of Webots PRO 5.8.0.