Cooperative Lifting of an Object Using a Hybrid Robotic System

Greg Brown
CSCI 7000-06: Multi-Robot Systems
University of Colorado at Boulder
browngp@colorado.edu

Abstract— **We present a hybrid system allowing multiple robots to cooperate in lifting and moving an object of arbitrary geometry. The robots navigate by alternating between different gradient navigation functions to evenly distribute themselves around the object, then follow a gradient to move in formation without dropping the object. The paper presents modeling results of the system using a discrete-time simulation of the robots and a probabilistic model of the overall system.**

I. INTRODUCTION

Using multiple robots to solve a problem can improve the scalability and robustness of the system. In this paper we examine a system for cooperatively lifting and moving an object of arbitrary geometry. A user places a beacon (perhaps RF-ID) on the object to be moved, and a second beacon at the object's destination. Some number of robots will then converge on the object, distribute themselves evenly around the object, and then move the object together to its destination. There are two key phases to this problem: (1) approaching and evenly distributing the robots around the object (the "Approach" phase), and (2) keeping the robots in formation as they move the object so it is not dropped (the "Move" phase).

The Approach phase has been examined in many other domains such as grasping objects with robotic arms[1], "caging" objects to push them[2], and maximizing the forces that can be applied to a barge by tugboats[3]. All of these problems involve trying to achieve an optimal "grasp" on the object so that it can be properly manipulated. In our case of lifting and moving an object the optimal grasp requires evenly distributing the robots around the object to prevent dropping the object. We assume that the object in question is of uniform density so that having more robots on one side or another is not a consideration. Furthermore, the shapes used in our simulations are all symmetric geometric shapes rather than shapes with more irregular patterns.

Our system approaches and grasps the object by each robot alternating between converging on the object and being repelled by the other robots. Gradient navigation functions (based on the work of Rimon & Koditschek[4] and Tanner & Kumar[6]) guide the robots' movements and prevent them from colliding with each other.

The Move phase requires the robots to maintain the positions they have reached relative to one another and relative to the object. Other work has looked at flocking[5] or moving robots in formation[6], but does not seem to have dealt with issues where getting out of formation would result in a catastrophic event (such as dropping the object).

The move phase also uses a gradient navigation function. The gradient balances the need for the robots to stay in formation around the object with the pull towards the object's destination.

We test our system using a discrete time simulation of the robots and compare the results to a probabilistic model of the convergence of the approach phase of the system.

II. SIMULATION DESCRIPTION

Figure 1 depicts the state machine for each of the robots in the system. The robots have access to the following sensor information:

- Bump sensor triggered by running into object.
- Direction and distance to the two beacons.
- Direction and distance to all other robots.
- A signal to indicate all the robots are ready to move the object.

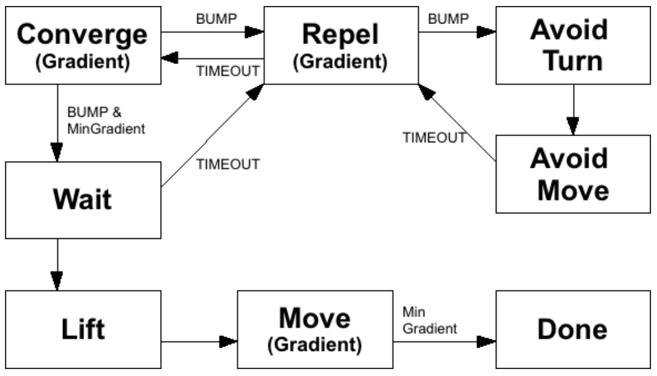

Fig. 1. State machine for a single robot. Robot begins in the $Converge$ state.

A. Approach Phase

The object is approached by alternating between the $Converge$ and $Repel$ states. In the $Converge$ state robots follow a gradient that pulls them towards the beacon that has been placed on the object. Equations 1, 2, and 3 show the navigation function that the robots use in converging on the object. In addition to the pull towards the object (γ) , robots avoid each other when they get too close together (β).

$$\gamma = \|r_i - r_{obj}\| \tag{1}$$

$$\beta = \prod_{j=1}^{N}(1 - \frac{1+d_c^4}{d_c^4}\frac{(\|r_i - r_j\|^2)^2}{(\|r_i - r_j\|^2 - d_c^2)^2 + 1})^{\frac{sign(d_c - \|r_i - r_j\|)+1}{2}} \tag{2}$$

$$NavigationCost = \frac{\gamma^2}{(\gamma^{\kappa_c} + \beta)^{1/\kappa_c}} \tag{3}$$

By following the gradient of the $NavigationCost$ function the robots will converge and bump into the object they are trying to lift while avoiding running into each other. Figure 2 shows a three dimensional plot of an example navigation function.

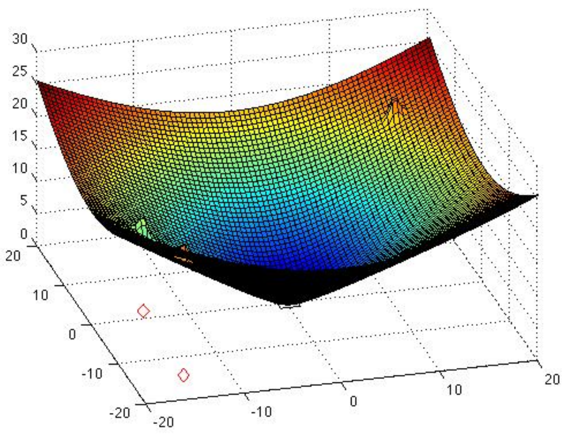

Fig. 2. Converge gradient. Red diamonds indicate location of other robots.

Once a robot detects that it has bumped into the object it transitions into the $Repel$ state. In this state the robot follows a navigation function which pushes the robots away from each other. Equations 4 and 5 govern the gradient that the robot follows in this state. The repel function (β) falls off as a robot gets further from all of the other robots.

$$\beta = \prod_{j=1}^{N} \|r_i - r_{obj}\|^2 - d_r^2 \tag{4}$$

$$NavigationCost = \frac{1}{(1 + \beta)^{1/\kappa_r}} \tag{5}$$

Following the gradient of this cost curve will result in the robots dispersing themselves evenly around the area centered on the object. Figure 3 shows a plot of an example $Repel$ navigation function.

A robot reaches the "optimal" location on an object when it bumps into that object in the $Converge$ state and the $Repel$ gradient indicates the robot is to move directly away from the object's beacon. At this point the robot stops and waits for all of the other robots to signal they are also done. If the other robots are not ready within T_{wait} then the robot will transition back to the $Repel$ state to try and find a better position. This helps avoid any potential deadlock case where the geometry of the object causes multiple robots to block another from reaching the object.

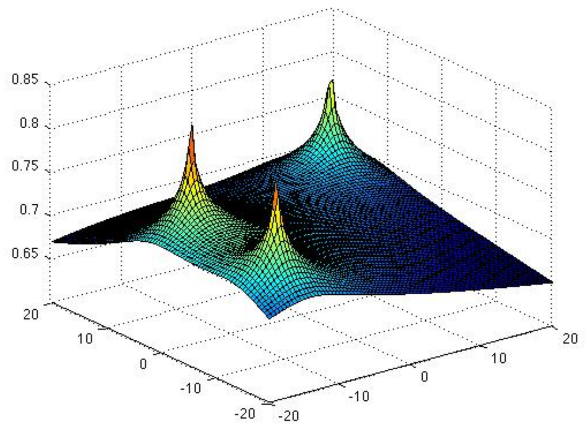

Fig. 3. Repel gradient. The peaks corresponds to the locations of other robots.

B. Move Phase

Once the robots have all indicated they are ready to move the object they lift the object. For this project we have assumed that lifting the object works immediately and without failure. We also assume that the robots have a forklift mechanism that can rotate 360 degrees as the robot turns. Future work could use a more limited rotation of the forklift by allowing the robots to move backward towards the object's destination.

After lifting the object a robot enters the MOVE state where it follows a gradient towards the object's destination beacon. To ensure that the robots do not move too far away from the object, they only move in short steps (r_{γ_i} and then use the object beacon's position to determine when the other robots have also moved forward. The robot uses the initial distance vector (r_{ideal_i}) to the beacon to "communication through the object." As the robot moves forward it is held back by r_{ideal_i}. Equations 6 and 7 show the vector to the point where the robot is moving and the goal function associated with it.

$$r_{\gamma_i} = r_{ideal_i} + d_m \frac{r_{ObjCenter} - r_{ObjDest}}{\|r_{ObjCenter} - r_{ObjDes}\|} \tag{6}$$

$$\gamma_1 = \|r_i - r_{\gamma_i}\| \tag{7}$$

Although the above equations do pull a single robot towards the destination, there are three interrelated problems which can cause the orientation of the robot to vary widely:

1) The error in the sensing and actuation of the robot means the robot may not head directly for r_{γ_1}.
2) The discrete nature of the system: the robot will decide to move in a direction and potentially travel past r_{γ_1}.
3) The robot will spiral towards r_{γ_1} as it waits for the other robots to catch up.

If the robots are allowed to spin they will rarely all be pointed in the direction of the destination and the object will make very slow if any progress.

We used two solutions to address this problem. First, the robot must be able to move backwards. This allows the robot to overshoot the r_{γ_1} point and not have to change its orientation to correct the overshoot. This however is not sufficient. Since the robot will often end up moving not perfectly in line with the intended direction of travel, even a backwards movement may be accompanied by a change in orientation. This hopping back and forth around r_{γ_1} will also cause the robot to spin.

To prevent the robot from spinning, we have also added a valley gradient (Equation 9) that is perpendicular (see slope Equation 8) to the intended direction of travel.

$$m = -\frac{r_{ObjCenter_x} - r_{ObjDest_x}}{r_{ObjCenter_y} - r_{ObjDest_y} + .0001} \tag{8}$$

$$\gamma_2 = \frac{|mr_{i_x} - r_{i_y} - mr_{gamma_x} + r_{\gamma_y}|}{\sqrt{m^2+1}} \tag{9}$$

$$NavigationCost = \gamma_1^{\kappa_{m1}} \gamma_2^{\kappa_{m2}} \tag{10}$$

Now when the robot follows the combined gradient (Equation 10) if the robot happens to move beyond its next intended step it will move backwards but not change its orientation very significantly. The gradient field depicted in Figure 4 better illustrates the combined navigation function.

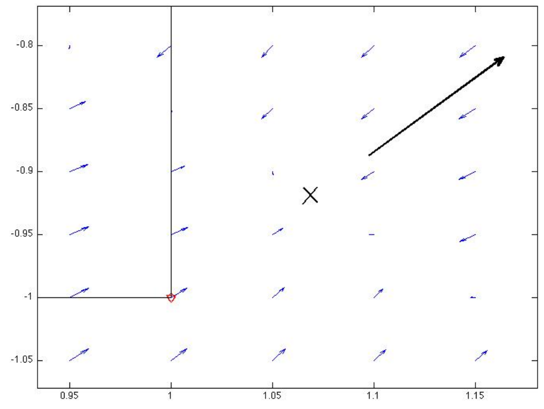

Fig. 4. Move gradient. The "X" marks location of next step in gradient. The arrow points in direction of the object's destination.

III. Model Description

To better verify the convergence of approaching the object before moving it we created a probabilistic model of the system to compare against the discrete-time simulations. The model represents the system as a series of states with some expected number of robots ($\hat{N}$) in each of the states. Transitions between states represent the expected number of robots to transition between the states on each time step of the model. Figure 5 shows the state diagram for the model. When all of the robots have reached the final $Wait_N$ state in the model, all robots in a real system should, on average, have completed approaching the object.

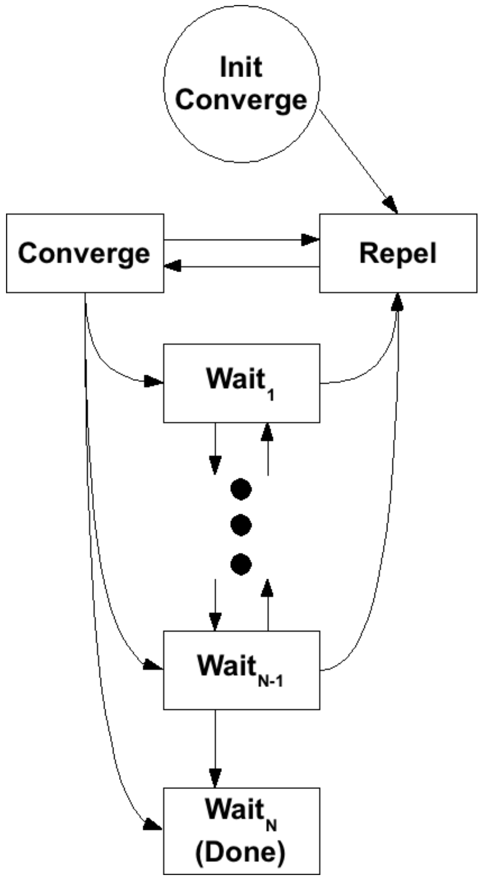

Fig. 5. State transition diagram for the system model. Each state transition is governed by a difference equation.

Below we present the difference equations that describe the transitions between the states and then we will discuss the parameters governing the model.

$$\hat{N}_{InitConv}(t+1) = \hat{N}_{InitConv}(t) - \frac{\hat{N}_{InitConv}(t)}{T_{InitHit}} \tag{11}$$

$$\begin{aligned}\hat{N}_{Conv}(t+1) = \hat{N}_{Conv}(t) &+ \frac{\hat{N}_{Repel}(t)}{T_{Repel}} \\ &-(1-P_{min})\frac{\hat{N}_{Conv}(t)}{T_{Hit}} \\ &- \sum_{i=1}^{nBots} \frac{P_{Min}P_{Wait_i}\hat{N}_{Conv}(t)}{T_{Hit}}\end{aligned} \tag{12}$$

$$\begin{aligned}\hat{N}_{Repel}(t+1) = \hat{N}_{Repel}(t) &+ \frac{(1-P_{Min})\hat{N}_{Conv}(t)}{T_{Hit}} \\ +\frac{\hat{N}_{InitConv}(t)}{T_{InitHit}} &- \frac{\hat{N}_{Repel}(t)}{T_{Repel}} + \sum_{i=1}^{nBots}\frac{\hat{N}_{Wait_i}(t)}{T_{Wait}}\end{aligned} \tag{13}$$

$$\begin{aligned}\hat{N}_{Wait_i}(t+1) = \hat{N}_{Wait_i}(t) + \frac{P_{Min}P_{Wait_i}\hat{N}_{Conv}(t)}{T_{Hit}} \\ + \frac{P_{Min}\hat{N}_{Wait_{i-1}}(t)\hat{N}_{Conv}(t)}{T_{Hit}} - \frac{\hat{N}_{Wait_i}(t)}{T_{Wait}} \\ - \frac{P_{Min}\hat{N}_{Wait_i}(t)\hat{N}_{Conv}(t)}{T_{Hit}} + \frac{\hat{N}_{Wait_{i+1}}(t)}{T_{Wait}} \\ - \frac{\hat{N}_{Wait_i}(t)}{T_{Wait}}\end{aligned} \quad (14)$$

The model looks very similar to the state diagram for the individual robots with a few exceptions. (1) We have removed the AVOID state under the assumption that it rarely occurs and is just an extension of the $Repel$ state. (2) A separate starting state was added for the initial convergence. This state represents the fact that the robots start with random positions in the arena whereas the normal convergence state occurs after a fixed number of time-steps in the $Repel$ state. (3) There are multiple $Wait_i$ states depending upon how many of the robots are waiting for the other ones to complete. For any single robot it can transition to the neighboring $Wait_{i-1}$ or $Wait_{i+1}$ state when another robot transitions from $Wait_i$ to $Repel$ or from $Converge$ to $Wait_{i+1}$ respectively. A robot in the $Wait_i$ state can itself transition to the $Repel$ state. The final $Wait_N$ state can only be transitioned into and never out of since this is the final goal state of a robot: to have all of the other robots also ready to lift the object.

Many of the parameters in the above equations are shown in Table III, but some of the more complicated ones are shown below. The probability (P_{Wait_i}) of which $Wait_i$ state a robot transitions into is given by Equation 15.

$$P_{wait_i} = \begin{cases} 1 & \sum_j \hat{N}_{Wait_j}(t) = 0, i = 1 \\ 0 & \sum_j \hat{N}_{Wait_j}(t) = 0, i \neq 1 \\ \frac{\hat{N}_{Wait_i}(t)}{\sum_j \hat{N}_{Wait_j}(t)} & \text{otherwise} \end{cases} \quad (15)$$

The probability is dependent on how many robots are currently in the other $Wait$ states since a robot can't transition into $Wait_3$ if there are no robots in $Wait_2$.

The average time that a robot stays in the $Converge$ state depends on how far away from the object it is and the speed the robot travels at. Equation 16 gives the average time to converge assuming that 75% of the time the robot moved in reverse during the $Repel$ state and 25% of the time it turned and moved away from the object. The combination of the two are needed to account for the time for the robot to turn and the difference between the forward and reverse speeds. This parameter is an estimation of the time in the $Converge$ state and could use further refinement.

TABLE I
PARAMETERS CONTROLLING THE SYSTEM.

Parameter	**Description**	**Value**
$nBots$	Number of robots in the system	3-6
κ_c	Convergence Gradient sharpening factor	0.75
d_c	Convergence robot avoidance distance	1
κ_r	Repel gradient sharpening factor	3.25
d_r	Repel robot avoidance distance	0.1
κ_{m1}	Move γ_1 sharpening factor	0.5
κ_{m2}	Move γ_2 sharpening factor	0.75
d_m	Move gradient step distance	0.025
$T_{InitHit}$	Average time-steps to initially converge on object	262
T_{Repel}	Time robot is in $Repel$ state	100
T_{Wait}	Timeout for $Wait$ state	1000
T_{Turn}	Time for robot to turn 90 degrees	$90/5$
$\dot{r}_f$	Forward movement speed	0.05
$\dot{r}_b$	Backward movement speed	0.025
$Grad_{min}$	Minimum repel gradient to enter $Wait$ state	0.2

$$T_{Hit} = \frac{0.75 \times \dot{r}_f(T_{Repel} - T_{Turn}) + 0.25 \times \dot{r}_b T_{Repel}}{2\dot{r}_f} \quad (16)$$

$$P_{Min} = \frac{3}{nBots} \quad (17)$$

Equation 17 represents the probability that a robot finds a minimum $Repel$ gradient when it bumps into the object, which would cause the robot to go into the $Wait$ state. This value is certainly inversely proportional to the number of robots in the system since as the number of robots increases a single robot will have more difficulty finding an even spacing between the robots. The numerator of three however was picked through comparison with the simulation, and as such this parameter is an estimation of the probability.

IV. RESULTS

The discrete-time simulation was run 40 times with 3, 4, 5, and 6 robots on four different shapes (square, star, rectangle, triangle) for a total of 640 simulation runs. Figure 6 shows a histogram of the time to complete approaching and moving the object on all of these runs for the different number of robots. For five and six robots the simulation does not always complete moving the object within 5000 time-steps. We believe this is due to needing to further optimize some of the parameters which determine when a minimum $Repel$ gradient has been reached.

The model's results are presented in Figure 7 in comparison to the percentage of simulation runs that completed the approach phase by that misstep. For the five and six robot cases the model fairly accurately tracks the simulation results. The model is not as accurate for the three and four robot cases which initially take longer to start converging in simulation and then all converge very quickly around 1000 time-steps. The discrepancy is most likely due to the problem of modeling a discrete time system with continuous functions. Nevertheless, the model does give a fairly good approximation of the convergence dynamics of the system.

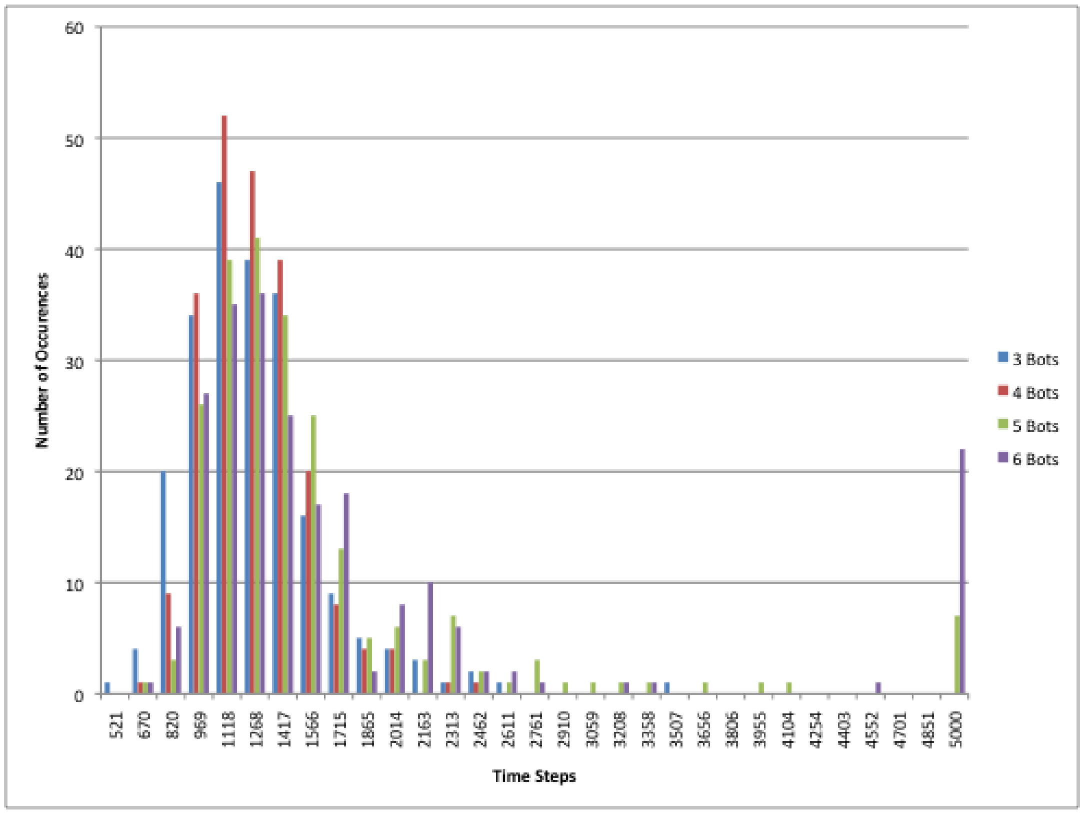

Fig. 6. Histogram showing the time to complete approaching and moving the object in discrete-time simulation.

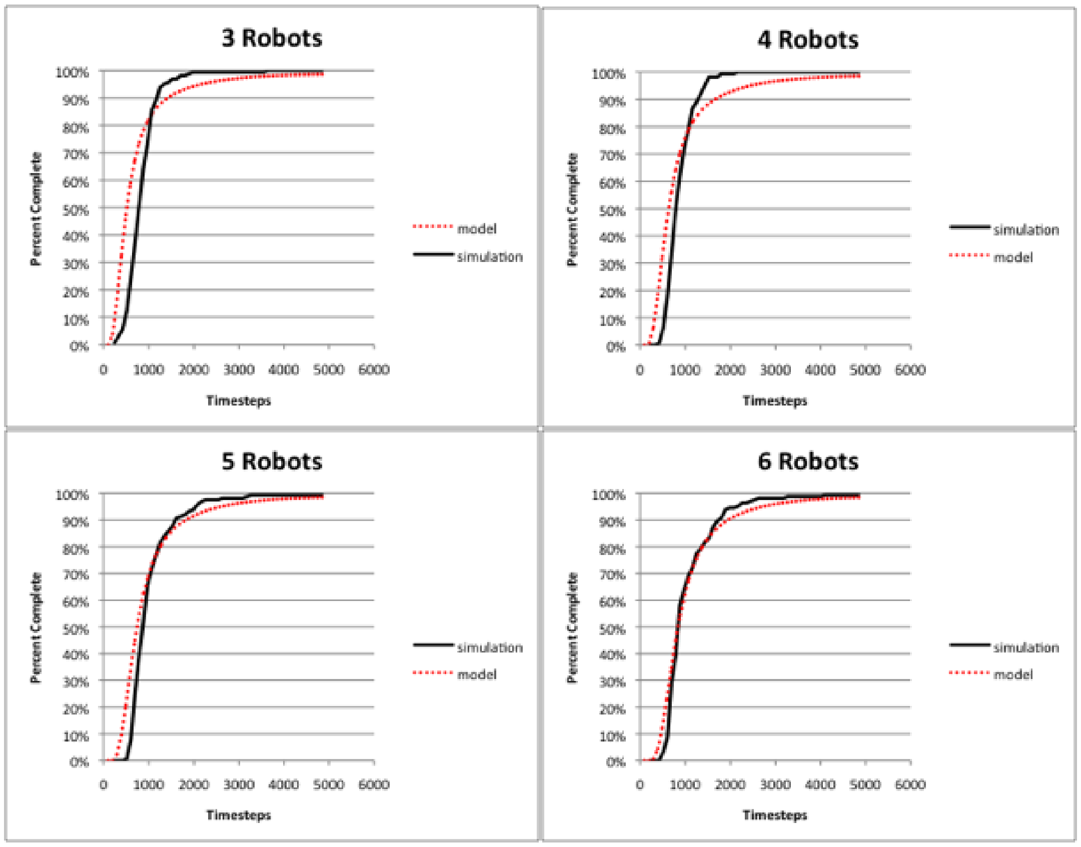

Fig. 7. Comparison between model and simulation results for different numbers of robots.

V. CONCLUSIONS

We have presented a hybrid multi-robot system for cooperatively lifting and moving an object using gradient navigation functions. The system was examined using a discrete time simulation and a probabilistic model. Both methods suggest that a system based on these methods would be feasible for implementation in a real system.

VI. FUTURE WORK

Some additional work still remains in looking at optimizing the parameters of the system through simulation and modeling. Additional modeling of what causes an object to be dropped should be examined by creating a model of the movement phase of the system. This may lead to improvements in how the robots arrive at an optimal ”grasp” on the object. The number of robots needed to lift an object could also be examined so that extra robots that are not really needed to move an object can be reallocated to other tasks. The actual lifting task has also not been addressed by the current research.

Finally, this system needs to actually be implemented in real hardware to determine the extent to which noise in the sensing and actuation would affect the algorithms. The current system simulation has very little of the noise that a real system would have to cope with. Although some of the noise could be simulated, many of the noise sources are dependent on how an actual system would be built, and are thus difficult to fully predict ahead of time. Moving ahead with testing on a real system and feeding the results back into these models would be the fastest method of gaining accurate insight into the impact of these noise sources.

REFERENCES

[1] Liu et al. On quality functions for grasp synthesis, fixture planning, and coordinated manipulation. IEEE International Conference on Robotics and Automation (2004)

[2] Fink et al. Multi-robot manipulation via caging in environments with obstacles. IEEE International Conference on Robotics and Automation (2008)

[3] Esposito et al. Cooperative manipulation on the water using a swarm of autonomous tugboats. IEEE International Conference on Robotics and Automation (2008)

[4] Rimon and Koditschek. Exact robot navigation using artificial potential functions. IEEE transactions on Robotics and Automation (1992)

[5] Lindh et al. Flocking with obstacle avoidance: A new distributed coordination algorithm based on voronoi partitions. IEEE International Conference on Robotics and Automation (2005)

[6] Tanner and Kumar. Formation stabilization of multiple agents using decentralized navigation functions. Robotics: Science and Systems (2005)