Emergence of Straight-Line Trajectory in Cooperative Object Transportation

Peam Pipattanasomporn, Sorawish Dhanapanichkul, and Attawith Sudsang
Department of Computer Engineering, Chulalongkorn University, Bangkok 10330
{peam.p,sorawish.d,attawith}@cp.eng.chula.ac.th

Abstract

Several tasks cannot be accomplished by a single robot working in isolation. Obvious examples include transportation of large objects. Multiple robot cooperation is therefore a natural choice for overcoming limited capabilities of an individual robot. This paper addresses the problem of designing a distributed control strategy for cooperative transportation of an object by a team of mobile robots. We investigate a new approach for which the object is simply placed on top of the robots and let the robots function as if they are the driving wheels of the object. We present a simple distributed algorithm that enables the robots to carry an object along a straight path without any explicit communication. The synchronization of the robots emerges as an outcome of the interaction between each robot and the transported object. Simulation experiments are presented with results showing robustness under varying sensory noise condition and scalability on varying team size of upto 100 robots.

1 Introduction

Moving an object from one place to another was among a few earliest applications in robotics. A large number of industrial robots today are involved in transportation of material in factory setting. How an object can be transported depends largely on its size and weight. A common practice is to attach part of the robot with the object (e.g., by grasping) so that the object can be moved together with the robot. Examples include the use of a robot arm equipped with a gripper or a suction device (Fig. 1 (a).) With this approach, a given robot can handle objects no larger than a certain size; objects larger than the limit need to be handled by a bigger or more complex robot. Building a large complex monolithic robot is, however, not an attractive solution due to high cost and level of technology required. A natural alternative is to deploy a team of simple robots. A few methods for multi-robot object transportation have been proposed in the area of cooperative robotics. In this paper, we propose a new cooperative approach for an object carrying task.

Our approach is conceptually equivalent to temporarily adding wheels to a stationary object. The object to be transported is simply placed on top of a team of homogeneous mobile robots (Fig. 1 (b)). With friction between the object and the supporting robots, motion of the robots will cause the object to move. In principle, desired motion of the object can be produced by appropriately controlling the robots. Unlike other works where several robots help grasp an object while moving, our approach does not need any grasping device and hence has no need to monitor or control grasping forces. Support for the object is directly provided by the robots. In this paper, the power of this cooperative carrying paradigm is not yet explored in its full generality. More precisely, limiting

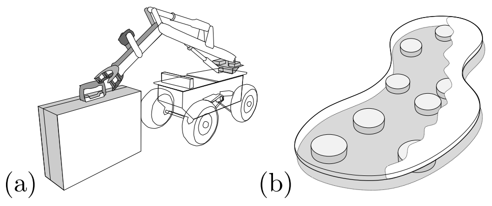

Figure 1: (a) a mobile manipulator is awkwardly picking up a luggage (b) setup showing mobile robots supporting the plate to be transported

that the robots are not allowed to communicate and do not have the knowledge of their configurations in the workspace, we will present a distributed control algorithm that enables the robots to cooperatively carry a plate along a straight-line trajectory. The proposed control algorithm decides how each robot should move based on the information from its local sensor measuring how the object slips relative to the robot's body. This is done without requiring any model of the object or any knowledge of the dynamics. Although the direction of the resulting trajectory cannot be dictated, translational motion of the object emerges very fast and robustly. According to simulation results, the approach exhibits great robustness and scalability. We present several simulation experiments showing successful straight-line transportation under varying sensory noise condition and team size.

Several methods have been proposed in robotics literature for multi-robot object transportation. The common reason is usually that it makes more sense to deploy many existing simple robots than to build a new powerful one [2]. Several works are based on the notions of force and form closure [6, 3, 4, 10]. In these works, a robot is modeled as an agent that induces a contact with the object. Object transportation can then be thought of as a grasping task, i.e., having multiple robots grasp the object and take it from place to place by maintaining a rigid grasping formation. These methods usually need to closely sense and control the grasping forces so that force closure condition can be satisfied during the entire transportation. Special force sensing devices are often needed. An interesting alternative is to apply the concept of caging [5]. With caging, an object is transported by being trapped (not grasped) in a moving formation of robots. Some robots simply push the object while the others guarantee that the object cannot escape from the formation. This approach relaxes the need for the robots to constantly keep contact with the object. It does not require as precise synchronization as grasping based approaches. An obvious drawback is that, at a given moment, only few robots can help pushing while many are left idle only to satisfy the caging condition. Examples of caging based approaches include [5, 11, 9]. Communication is an important issue in cooperative object manipulation. Inter-robot communication can be classified into either explicit or implicit [2]. Explicit communication refers to explicit act of information exchange while implicit communication is a by-product of the interaction between robots and environment. The approach presented in this paper is strongly based on implicit communication. It is shown in [1] that explicit communication is not necessary when an implicit form is available. It is also shown theoretically in [8] that the use of implicit communication can reduce the amount of explicit communication.

The rest of the paper is organized as follows. Sec. 2 presents the basic concept of the approach together with the proposed distributed robot control algorithm. In Sec. 3, the approach is verified and its performance is analyzed using several simulation experiments. Discussion about remaining issues of interest is given in Sec. 4 and a conclusion is provided in Sec. 5.

2 Methodology

As mentioned earlier, the key idea of the approach is to place the object to be transported on top of the robots and let the robots function as if they are the driving wheels of the object. In this paper, the approach is not yet investigated in its full generality: we assume that the object is simply a plate with some thickness and consider only the robots with the following specification: (1) every robot is in the same cylindrical shape with a flat top surface at the same height. (2) the drive system is omnidirectional, i.e., capable of translating in any direction. The motion command is in the form of the intended velocity (in the robot's frame). (3) Neither compass nor communication is available. (4) The top surface of each robot is equipped with a sensor for detecting how the object slips with respect to the robot.

Our objective is to develop a distributed algorithm that enables the robots to carry the object along a straight line trajectory. It is obvious that the object will be translated along a straight path when every robot moves in the same direction at the same speed. These conditions may be easily satisfied if the robots could explicitly communicate the common velocity. This is however not the case in our setting where communication and directional reference is not available. It is then natural to ask what can be accomplished without communication. Answers to this question is important since explicit communication is sometimes unreliable or even unavailable. Depending critically on communication often leads to problems in robustness and scalability.

Before presenting the proposed control algorithm, let us describe how everything works together. The idea is to program each robot with the control algorithm to be presented next. The robots are then randomly positioned such that the object being placed on top of them does not tip over (i.e., the center of mass of the object lies inside the convex hull of all the robots). Each robot is then allowed to move according to an initial random velocity. The object will be moved by the friction forces between the robots and the object. After this point, the control algorithm will take over; velocity of each robot will be updated repeatedly by the control algorithm in order to reduce the object's slippage. Once the object stops slipping, straight-line motion will emerge. It will be shown in Sec. 3 that the object's trajectory usually converges to a straight line very fast. The desired direction, however, cannot be dictated since no global reference is provided. We are currently investigating a setting for which some of the robots have the knowledge about the desired direction.

This idea will be discussed in Sec. 4.

We are now ready to present the proposed distributed control algorithm. It is described in the following *updateMotion* routine. Every robot runs the routine *updateMotion* once per tick cycle. In each run, the robot updates its intended velocity based on the currently commanded velocity and how the object slips. Variable *currentVel* represents the robot's currently commanded velocity vector and variable *slipVel* represents the vector of the object's slipping velocity (with respect to the robot). Both velocities are expressed in the robot's body frame. The velocity command to be executed next is obtained from a linear combination of *currentVel* and *slipVel* with constant coefficients C and D (line 3). Constants *MIN_SPEED* and *MAX_SPEED* are the minimum and maximum speed of every robot. Lines 4 and 5 ensures that the speed of the robot is within the specified range. Line 6 commands the robot to follow the newly computed velocity. Note that describing the object's slippage by linear velocity follows how the slippage sensor is modeled. In practice, a slippage sensor can be constructed by modifying a PC optical mouse [7]. This device essentially takes hundreds of pictures (of the sensed surface) per second and calculates the velocity of the mouse with respect to the surface based on the concept of optical flow.

```
1: $updateMotion(currentVel, slipVel)$
2:     $currentVel := C * currentVel + D * slipVel$
3:     **if** $currentVel > MAX\_SPEED$
           **then** $currentVel := MAX\_SPEED * \frac{currentVel}{|currentVel|}$
4:     **if** $currentVel < MIN\_SPEED$
           **then** $currentVel := MIN\_SPEED * \frac{currentVel}{|currentVel|}$
5:     $moveRobot(currentVel)$
```

An explicit function of the above simple algorithm is to align the robot's velocity with the object's slipping one. Collective effect of the specified reactive behavior results in the reduction of the object's slippage. This interaction between the robots and the object translates into the synchronization of the robots that allows the object to be carried along a straight path. The proposed algorithm does not need any model of the object and it does not make any assumption about the physics such as friction coefficients. The independence from modeling and communication makes the algorithm robust.

3 Simulation Experiments

The power of the control algorithm presented in the previous section can be best demonstrated by experiments. In this section, we present simulation experiments intended to examine various aspects of the approach under varying conditions. The simulator used in all experiments is written in C++ using the open source library Open Dynamics Engine (http://ode.org) for simulating dynamics. The simulation time step is set to 1 ms. Each robot is in a cylindrical shape with 0.2 m in diameter, 0.1 m in height and 2 kg in weight. Every robot is programmed with the control algorithm presented previously with $C = 1.6$ and $D = 1.4$. The maximum and minimum speed of each robot is set to 0.5 m/s and 0.2 m/s (resp.) The test object is a square plate with dimension 5 m $\times$ 5 m $\times$ 0.1 m and 30 kg in weight. Test objects in different plate shapes may be used as well but only the square plate is presented here since the scope of the paper does not cover varying geometry of the transported object. Unless specified otherwise, the static and kinetic friction coefficients between the robots and the object are set to 0.3. The velocity update rate (tick rate) is set to 10 Hz. Due to computing performance limitation, the driving system is not simulated from its working mechanism, e.g., motor components. To allow many robots to be simulated simultaneously, only approximate behavior of the driving system can be considered. Specifically, at the start of every simulation time step, the simulator's state containing the robot's velocity is set to the robot's currently commanded velocity. This allows the robot's configuration to be calculated based on external forces such as friction. Behaviors such as slip of wheels can be expressed with this technique. Despite its simplicity, the technique appears to be more realistic than assuming perfect control and actuation where the robot's velocity precisely follows the commanded velocity.

3.1 A Typical Successful Trial

Fig. 2(b) shows the trajectory of the test object from an experiment. The random initial position and direction of each robot are shown in Fig. 2(a). The position and direction of the robots when a straight-line trajectory is achieved is shown in Fig. 2(c). Observe that the relative position of each robot with respect to the object changes only slightly during the motion. The object is classified to move along a straight path when it maintains angular velocity smaller than 0.001 rad/s for longer than 1 seconds and during this period the linear speed is greater than 0.05 m/s. In this experiment, the object seems to gradually rotate counterclockwise during the first 4 seconds and then it constantly translates along a straight path. Detail of the motion is given in the plot of the object's orientation and speed in Fig. 3. This sample trajectory depicts a typical behavior of the object in successful cases. There are, however, situations for which the object seems to rotate indefinitely around a moving center. Such cases occur at a low rate. They will be discussed further in Sec. 4. Since the result from an

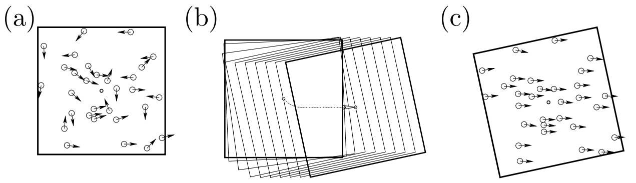

Figure 2: The trajectory of the object (see text)

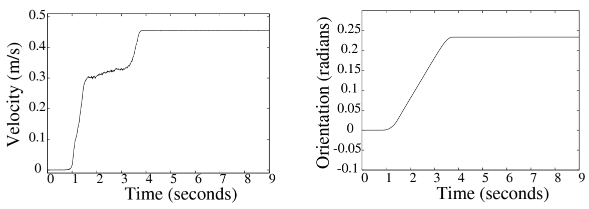

Figure 3: Velocity profile of the sample trajectory, from left to right: linear speed as a function of time, and orientation as a function of time

experiment depends on the initial random position and velocity of the robots, each experiment is performed several times (with different initial position and velocity) to collect average statistics: (1) convergence ratio, i.e., the frequency of trials the object ends up moving along a straight path within the first 2 minutes, and (2) convergence time, i.e., the average time required to achieve a straight path (averaging only over those trials achieving a straight path below 2 minutes). These two quantities is used throughout the rest of the paper to measure the performance of a certain experiment.

3.2 The Number of Robots

An important advantage of distributed approaches is the ability to easily add more agents into the tasks being performed. In our transportation task, a heavier object implies larger friction forces between the object and the robots; when the number of robots is fixed, the weight of the object can only be smaller than a certain limit or the friction will become too large for a robot to move. This is clearly confirmed by the plot in Fig. 4. Thirty robots are used in this experiment. The test object remains a square plate but varies its weight from 10 to 140 kg (step size of 10 kg). At each weight value, 50 trials are performed and statistics are collected. As shown in the plot, the convergence ratio drops drastically when the weight is increased from 80 to 110 kg. In this same range the convergence time jumps from 27 to 65 seconds. The task never succeed with the weight over 130 kg. Besides increasing the maximum payload

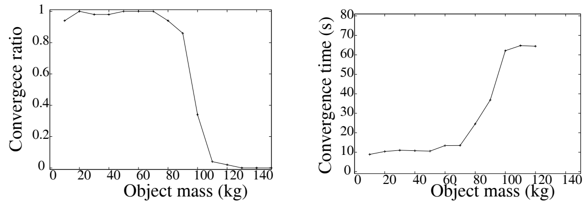

Figure 4: Convergence ratio and time as a function of the object's weight (with 30 robots)

capacity, it is interesting to ask whether the number of robots affect other qualities. Experiments are set up to investigate this issue. Varying number of robots are deployed in the experiments. For a given number of robots, 50 trials are performed. The convergence ratio and the convergence time from the experiments with varying number of robots are shown in Fig. 5. Note that when the number of robots is smaller than 10, the robots can rarely move; the object is simply too heavy and never achieves a straight path. Marginally, with 10 robots, straight path is achieved with convergence ratio 0.96 (48 out of 50 trials) and convergence time of 26.5 seconds. Increasing the number of robots to 15 sharply drops the convergence time to 10 seconds while still maintains the convergence ratio at 1.0. Increasing the number of robots further gradually reduces the convergence time and slightly reduce the convergence ratio. With 140 robots, 45 out of 50 trials still successfully achieve a straight path with average required time of 9 seconds. Besides demonstrating that the approach performs well with varying number of robots, data from the experiments interestingly suggest that when the number of robots exceed a certain number, the performance will not continue to be significantly improved. This number is clearly helpful in deciding how many robots should be deployed in a specific task. More research effort is needed to investigate this issue in greater depth.

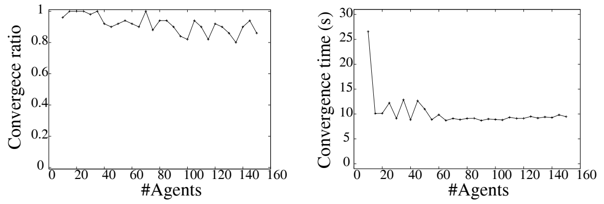

Figure 5: Convegence ratio and time as a function of the number of robots

3.3 Noisy Sensors

Up to this point, the slippage sensors are assumed to be perfect. This ideal condition rarely occurs in practice. It is therefore necessary that our approach is tested under the effect of noisy sensors. Since a slippage sensor returns the direction and speed of the slipping object, we model a noisy sensor by contaminating the sensed direction and speed with gaussian noise at varying standard deviations. Experiments with 50 robots are conducted. Fig. 6 plots the convergence ratio with respect to the standard deviation of the direction noise and the speed noise. As expected, the plot shows that the task always fails when the noise is larger than a certain level. What is surprising is that there exists a large zone where noise improves the convergence ratio. This zone is the flat region where the convergence ratio reaches 1.0 (the convergence ratio is 0.9 with perfect sensors). Although appropriate noise level may improve the convergence ratio, experiments show that the convergence time with noisy sensors is in general significantly longer than that of perfect sensors.

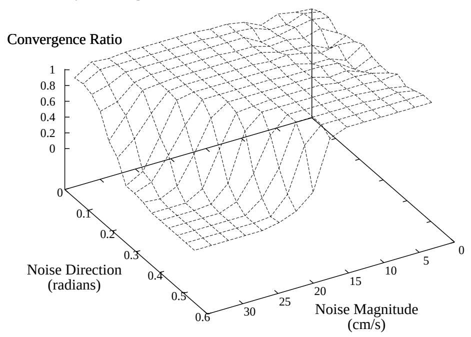

Figure 6: Results from experiments with noisy sensors plotting convergence ratio as a function of the standard deviation of direction noise and speed noise

3.4 Malfunction of Robots

It is unavoidable that robots sometimes fail to operate. It is therefore helpful to find out how the performance suffers when some robots malfunction. Experiments with 30 robots are conducted to investigate this issue. In this study, we consider that the robots completely stop moving when they malfunction. Varying number of robots are randomly chosen to become malfunctioned. The plots in Fig. 7 present the results of the experiments. We can see that the system sustains the inoperability of robots very well and only degrades gracefully when the number of malfunctioned robots is not larger than 5 (16% of team size).

4 Discussion

In this paper, all experiments are conducted in computer simulation. This allows experiments to be easily performed with various settings. In particular, we are able to investigate issues regarding a large number of robots which is impossible to study with real robots. Of course, the validity of the approach should also be confirmed with physical experiments. We are currently constructing an experiment platform using a team of 10 RoboCup style robots. Results from the experiments will be reported in our future papers.

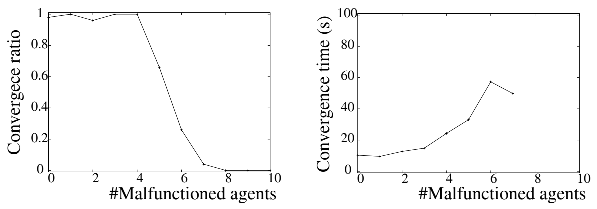

Figure 7: Convergence ratio and time as a function of the number of malfunctioned robots

As mentioned in Sec. 3, there are cases for which the object fail to achieve a straight path. The state of rotation when the object is being spun in a directions by the robots appears to be stable except that the position of each robot relative to the object gradually changes (it almost does not change when a straight path is achieved). This means that some robots may move out of the boundary of the object if the object is left in rotation for sufficiently long time. With simple sensors that can detect this out-of-bound situation, the control algorithm can be easily modified to always keep the robots under the object. Nevertheless, we does not only want to keep robots under the object, but also to take the object out of its rotation. An easy solution is to equip each robot with a compass. With a compass, the robot can track its heading and recognize whether it is stuck in a rotation state. To escape from such state, a simple method is to perform a reset: a new velocity is chosen at random and maintained for a short duration before giving the control back to the control algorithm. Our preliminary experiments show that this technique significantly increases the convergence ratio.

Interestingly, recognizing whether the object is stuck in a rotation state can also be done without using compasses. By observing a robot in a rotation state, it appears that most of the time the commanded velocity lies on one side of the slipping velocity (this does not apply when the object achieve a straight path). This means that the accumulated sum of the signed angle between the commanded velocity and slipping velocity will have its magnitude grow larger over time as the object rotates. Recognizing a rotation state can therefore be performed by comparing this sum with an appropriate threshold.

The control algorithm proposed in Sec. 2 achieves the robots' synchronization through implicit communication. This implicit communication is in the form of the interaction between the robots and the object. Besides facilitating implicit communication, the object may also be used as a medium for explicitly broadcasting simple messages. When the object is moved along a straight path, every robot tends to move in approximately the same direction (Fig. 2(c)). If one robot changes its heading abruptly, the object's trajectory will be affected which, in turn, results in the object's slippage that can be sensed by the other robots. This is an example of how a message can be broadcasted via the object. Now, let us assume that the object is being carried along a straight path and one robot is able to recognize which direction is heading to the target. The idea is that if the current direction is not acceptable, it is then possible for the robot to use the broadcasting technique described above to signal the other robots to change their directions. Provided that a direction changing algorithm (e.g., 10 degrees counterclockwise from the current velocity) is pre-programmed in every robot, the object will then be moved along a new straight path. This procedure can then be repeated until the object reaches its destination. We are studying this simple procedure in more detail and currently interested in deriving a technique that allows a more complex message to be broadcasted.

5 Conclusion

We have presented a distributed control algorithm that enables a team of robots to carry a plate along a straight path without explicit communication. The algorithm has been verified in several simulation experiments with results showing robustness and scalability . The beauty of the algorithm is its simplicity. The synchronization of the robots is collectively derived from the interaction between the robots and the object. Independence from modeling and communication avoids the inherent vulnerability to communication and modeling problems which results in a robust and scalable multi-robot object transportation method.

References

[1] T. Balch and R. C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):1–25, 1994.

[2] T. Balch and L. E. Parker. *Robot teams: from diversity to polymorphism.* A.K.Peters, 2002.

[3] M. Mataric, M. Nilsson, and K. Simsarian. Cooperative multi-robot box pushing. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 556–561, Pittsburgh, PA, August 1995.

[4] J. Ota, N. Miyata, and T. Arai. Transferring and regrasping a large object by cooperation of multiple mobile robots. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 543–548, 1995.

[5] E. Rimon and A. Blake. Caging 2D bodies by one-parameter two-fingered gripping systems. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, pages 1458–1464, Minneapolis, MN, April 1996.

[6] D. Rus, B. Donald, and J. Jennings. Moving furniture with teams of autonomous robots. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 235–242, 1995.

[7] V. Silva, F. Santos, and L. Almeida. A robust self-localization system for a small mobile autonomous robot. In *Proceedings of the ANIRob/IEEE Int. Symposium on Robotics and Automation*, 2002.

[8] D. J. Stiwell and B. E. Bishop. A framework for decentralized control of autonomous vehicles. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, 2001.

[9] A. Sudsang, F. Rothanger, and J. Ponce. Motion planning for disc-shaped robots pushing a polygonal object in the plane. *IEEE Transactions on Robotics and Automation*, 18(4):550–562, August 2002.

[10] T. Sugar and V. Kumar. Decentralized control of cooperating mobile manipulators. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, 1996.

[11] Z. Wang and V. Kumar. Object closure and manipulation by multiple cooperative mobile robots. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, 2002.