Distributed Centroid Estimation and Motion Controllers for Collective Transport by Multi-Robot Systems*

Golnaz Habibi[1], Zachary Kingston[2], William Xie [3], Mathew Jellins[4], and James McLurkin[5]

Abstract— **This paper presents four distributed motion controllers to enable a group of robots to collectively transport an object towards a guide robot. These controllers include: rotation around a pivot robot, rotation in-place around an estimated centroid of the object, translation, and a combined motion of rotation and translation in which each manipulating robot follows a trochoid path. Three of these controllers require an estimate of the centroid of the object, to use as the axis of rotation. Assuming the object is surrounded by manipulator robots, we approximate the centroid of the object by measuring the centroid of the manipulating robots. Our algorithms and controllers are fully distributed and robust to changes in network topology, robot population, and sensor error. We tested all of the algorithms in real-world environments with 9 robots, and show that the error of the centroid estimation is low, and that all four controllers produce reliable motion of the object.**

I. Introduction and Related Work

Multi-robot systems provide advantages in flexibility and robustness for object manipulation tasks that individual robots cannot provide. For example, in the event of a disaster, rescue missions often involve transportation of large numbers of large objects, multi-robot systems can transport these in parallel, using more robots for larger objects. A package-handling logistics center requires quick and efficient transportation of many objects of a wide range of sizes, shapes, and weights. Specialized manipulators for different object types could be costly to design and more complicated to implement than groups of simpler, highly scalable manipulators that can work together. In these applications, large populations of robots permit flexible task allocation and parallel operations, but require motion controllers to cooperatively move large objects.

We are interested in techniques that allow a group of simple robots to manipulate objects that are larger than any one robot can move alone, like the example in Fig. 1. Low-cost robots can be cheaply deployed in large populations, but often have limited computation and sensing, which requires different approaches for coordination.

A variety of different methods of multi-robot transportation have previously been studied. These studies can largely be separated into two different categories. One category has a high degree of communication between robots, and the other is more distributed, with very limited inter-robot communication. Le et al. [3] used a heterogeneous group of three robots in a leader-follower configuration to move an object, where a single leader robot had improved capabilities to coordinate object transportation using two follower manipulator robots. Object closure formation have been investigated for efficient multi-robot transport [4], [5]. Chen and Luh [6] approached the problem by dividing the coordination and control into five sub-tasks. Groß et al. [7], [8] used self-configurable robots and artificial neural network to transport a large object. Examples of work using limited communication use a swarm of decentralized micro robots to transport a large object without prior knowledge of the payload through common control [9] or by using force consensus [10]. These techniques are closest to our work.

Our approach is inspired by natural systems that are able to use large groups of agents for manipulation tasks [11]. The distributed manipulation they use offers advantages that a single manipulator cannot provide. The force applied to the object by individual agents can be more evenly distributed across the geometry of the object. Multiple agents on opposite sides of an object can generate a large torque, which

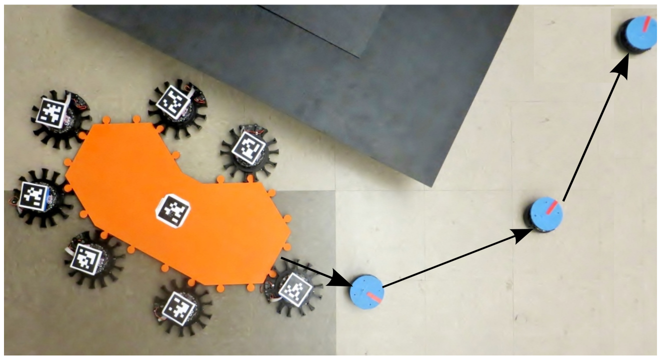


Fig. 1: Seven r-one robots with grippers (black circles) [1] are gripping an object (the orange "bean"). Our goal is to transport the object along the path marked by the guide robots (blue circles). In our previous work [2], we described algorithms that enable each guide robot to compute a collision-free pose for the object at the robot's location. This paper presents distributed controllers that use the guide robot's navigation information to control a group of manipulator robots. In this image, the manipulator robots will translate the object to the first guide, then rotate and translate to the second guide, and lastly translate again towards the third guide. This work presents distributed algorithms to estimate a shared center-of-rotation, the centroid of the object, and distributed motion controllers for translation, rotations, and combinations of the two.

*This work has been supported by the NSF under CNS-1330085

[1]Golnaz Habibi is a graduate student, Computer Science, Rice University, 6100 Main Street, 77005 Houston, TX golnaz.habibi@rice.edu
[2]Zachary Kingston is an undergraduate student, Computer Science, Rice University zkk1@rice.edu
[3]William Xie is a graduate student, Computer Science, University of Texas at Austin wxie@cs.utexas.edu
[4]Mathew Jellins is an undergraduate student, Computer Engineering, Purdue University mjellins@purdue.edu
[5]James McLurking is with the Faculty of Computer Science, Rice University jmclurkin@rice.edu

makes object rotation easier. Furthermore, the arrangement of multiple agents along the perimeter of the object improves the collective ability of the system to sense and avoid obstacles during transportation.

This paper presents distributed algorithms to transport objects of different sizes, weights, and geometries. Our algorithms are self-stabilizing, robust to dynamic network topology, changes in robot population, and sensor errors. We aim to address situations where global sensing, communication, and geometry is not readily available or too costly to implement. We use multi-hop communications [12] to exchange local information *and* local geometry in order to cooperate with other robots, avoiding the need for a shared global coordinate frame [13]. We present a novel, tree-based algorithm to accurately estimate the centroid of the object. Centroid estimation had been accomplished using consensus or gossip-based algorithms in previous literature [14], but this approach has limitations that we overcome. We describe four different motion controllers for the object: rotation around a pivot robot, rotation around the centroid of the object, translation towards a guide robot, and a combination of rotation and translation (See Fig. 5). The motion controllers allow the robots to manipulate an object with high dexterity, and collectively cover all types of motion on the plane. We implement and verify the centroid estimation and controller effectiveness in a physical robot system of 2-9 robots.

II. Model and Assumptions

We assume that our robots are inside an environment too large for centralized communication. A communication network is built by the robots using inter-robot communications between nearby robots within a fixed distance d, where d is much smaller than the size of the environment. We can model the robot's communications network, $G = (V, E)$, as an undirected unit disk graph. Each robot constitutes a vertex $u \in V$, where V is the set of all robots and E is the set of all robot-to-robot communication links. The set V_m is the set of all manipulator robots. The neighbors of each vertex u are the set of robots within line-of-sight communication range d of robot u, denoted $N(u) = \{v \in V \mid \{u, v\} \in E\}$. We assume that G is connected.

Our robots are homogeneous and are modeled as a disk and move using non-holonomic differential drive. Each robot u has a unique id, $u.id$. Each robot is situated at the origin of its own local coordinate frame with the $\hat{x}$-axis aligned with its current heading. Robots can measure the relative pose of its neighbors (See Fig. 2).

We model algorithm execution as proceeding in a series of discrete *rounds*. While the robots actual operation is asynchronous, implementing a synchronizer simplifies analysis greatly and is easy to implement [13].

We assume there is a large population of robots spread over the environment [15]. In previous work, we developed algorithms for these *guide* robots to generate the configuration space and plan a path for an object to be transported [2]. A small subset of the robot population are *manipulator robots* are responsible for transporting the object, and are

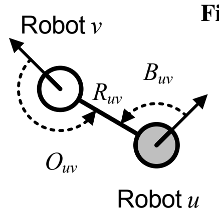


Fig. 2: Local network geometry of robot v measured from robot u. B_{uv} is the bearing, the angular position of robot v from robot u's heading. O_{uv} is the orientation, the relative heading of robot v from B_{uv}, and R_{uv} is the distance between the two robots.

attached to it. In this work we assume the manipulator robots can communicate with one guide robot to determine the local motion required. The manipulator robots have no prior knowledge of the shape or size of the object. We model the robots' attachment to the object as pin joints, which means that an individual robot can apply a force to the object, but not a torque. Multiple robots can rotate and translate objects by exerting forces in cooperation, generating net torques and forces on the object.

III. Estimation of Centroid of the Object

Three of our motion controllers require the robots to compute a shared center of rotation. We assume that there are enough robots surrounding the object such that their positions form a polygon that resembles the shape of the object. As the centroid of a polygon is located at the average positions of all of the objects vertices, the *object's* centroid can be approximated by taking the average of the *robot's* positions.

In order for our controllers to be effective, the centroid estimation needs to be robust to the types of errors found in multi-robot systems. Previous approaches [14] present a consensus-based approach. This type of estimate is accurate, uses limited communications, and requires only local information. However, these algorithms compute consensus of the initial configuration of the network. Any initial error will be maintained indefinitely. Consensus-based algorithms will converge within acceptable error when communication failure is present, but these algorithms are not usually robust to changes in robot configuration and are not self-stabilizing [16].

We use a distributed, tree-based algorithm to estimate the centroid. Each robot $u \in V_m$ builds a tree T_u rooted on itself, extending to all manipulator robots, shown in Fig. 3. The centroid of the manipulator robots is $x_c = \frac{\sum u.x}{|V_m|}, y_c = \frac{\sum u.y}{|V_m|}$. Our approach distributes the sum computation so that each robot computes the number of children on its subtree, and the sum of their positions *in its coordinate frame*. In this way, the messages propagated up the tree are always of constant size, and the root of the tree can perform the final division by the total number of children as the final step.

Each robot needs to relay the trees for every other manipulator robot, which requires each robot to maintain a list of these trees, labeled by the ID of the source robot. Building a consistent list of trees on each robot requires a list of all manipulator robots, which we build with the *Extreme-Comm* algorithm [17]. These trees are communicated to all robots each round, this requires $O(|V_m|)$ messages per robot per round, which is only viable for small values of $|V_m|$.

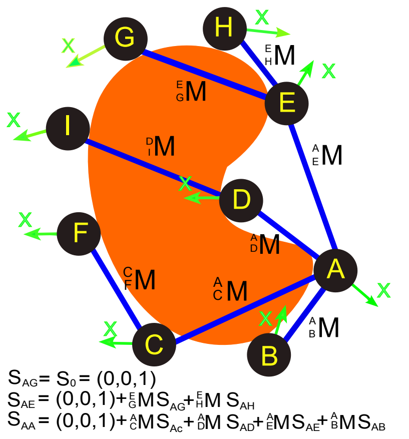

G
H
E
I
D
F
A
C
B
x
$^{E}_{H}M$
$^{E}_{G}M$
$^{D}_{I}M$
$^{A}_{E}M$
$^{A}_{D}M$
$^{C}_{F}M$
$^{A}_{C}M$
$^{A}_{B}M$
$S_{AG} = S_0 = (0,0,1)$
$S_{AE} = (0,0,1) + {^{E}_{G}M} S_{AG} + {^{E}_{H}M}\, S_{AH}$
$S_{AA} = (0,0,1) + {^{A}_{C}M} S_{Ac} + {^{A}_{D}M}\, S_{AD} + {^{A}_{E}M} S_{AE} + {^{A}_{B}M} S_{AB}$

Fig. 3: A tree rooted on robot A. Each node receives the sum its subtree's nodes positions. The node converts the positions to its local coordinate frame. This node then calculates its sum and passes it up to its parent. The root of the tree then calculates the centroid by diving its sum by the total number of children. There are trees rooted on every node in the network that are calculated concurrently.

Algorithm 1 Algorithm for Centroid Estimation

```
1:  $\vec{S_0} \leftarrow (0,\ 0,\ 1)$
2:  **Repeat forever on each robot** $u$
3:  **for all** $T_i \in \{T_1, \dots, T_n\}$ **do**
4:      $\vec{S_{iu}} \leftarrow \vec{S_0} + \sum_{v \in N_c(u)} {^{u}_{v}M} \vec{S_{iv}}$
5:  **end for**
6:  **if** $u$ root of $T_i$ **then**
7:      $X_{sum} \leftarrow \vec{S_{iu}}.x$
8:      $Y_{sum} \leftarrow \vec{S_{iu}}.y$
9:      $C_{count} \leftarrow \vec{S_{iu}}.cc$
10:     $Centroid_i \leftarrow (X_{sum}/C_{count}, Y_{sum}/C_{count})$
11: **end if**
```

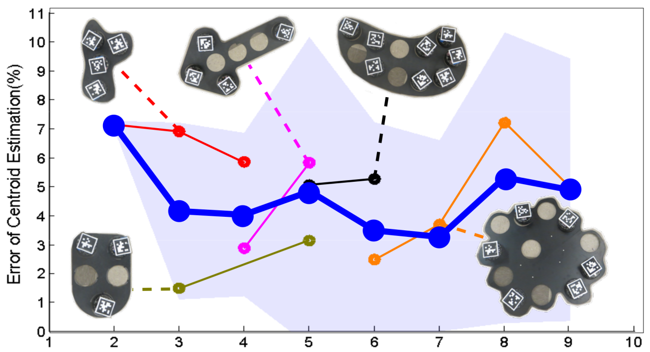
In a tree T_i, robot u reads the message $\vec{S_{iv}} = (S.x, S.y, S.cc)$ from each of its children, $v \in N_i(u)$, where $S.x$ and $S.y$ are the sum of the positions of the descendant nodes of v, and $S.cc$ is the total size of v's subtree. In line 4 of the Algorithm 1, robot u calculates and broadcasts its own value, S_{iu}. The position of child robot v is transformed to the local coordinate frame of its parent u using the transformation matrix, ${^{u}_{v}M}$ defined in (1). The parameter θ for the transform is given by (2).

$$ {^{u}_{v}M} = \begin{bmatrix} \cos(\theta_{vu}) & -\sin(\theta_{vu}) & x_{vu} \\ \sin(\theta_{vu}) & \cos(\theta_{vu}) & y_{vu} \\ 0 & 0 & 1 \end{bmatrix} \tag{1} $$

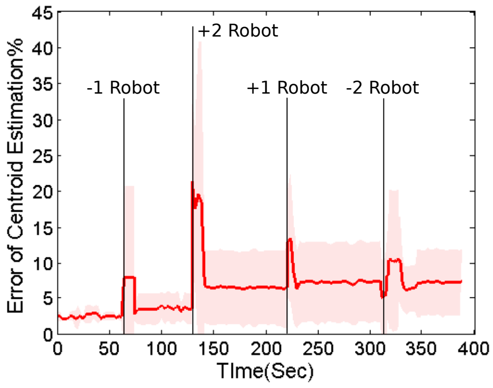
$$ \theta_{vu} = \pi - O_{vu} + B_{vu} \tag{2} $$

Using algorithm 1, each robot u calculates the sum of position of its subtree. To show how this algorithm works, consider the tree of robots rooted on A (See Fig. 3). If each robot passes the position of its children to its parent, robot A, the root, would have the positions of all children in the tree. We define the position of a robot u in robot v's coordinate frame to be ${^{u}v}$. ${^{u}u}$ is $(0,\ 0,\ 1)$. Consider the subtree of A consisting of the set of robots $\{A, E, G, H\}$. The sum of the positions of the robots in this subtree at robot A would be

$$ \begin{aligned} S &= {^{A}_{E}M}{^{E}E} + {^{A}_{E}M}{^{E}_{H}M}{^{H}H} + {^{A}_{E}M}{^{E}_{G}M}{^{G}G} \\ &= {^{A}_{E}M}({^{E}E} + ({^{E}_{H}M}{^{H}H} + {^{E}_{G}M}{^{G}G})) \end{aligned} $$



(a) Centroid measurement error for many types of objects



(b) Centroid Self-Stablization

Fig. 4: **(a)** Data on centroid estimation. The mean error for the estimate is 4.64%. The blue line shows mean error over all objects. The standard deviation is shown by the blue shading. The other colored lines show mean error for different objects. **(b)** Data from the self-stabilization experiment. Starting with 5 robots, robots were added and subtracted from the population in different locations over time. The estimate stabilizes to a steady result a few rounds after each disturbance.

A. Analysis and Experimental Results

We tested our algorithm for centroid estimation on 2-9 r-one robots, on 5 different objects, for a total of 90 trials. The results are shown in Fig. 4a, which illustrates the error of the centroid estimation to the actual location of the centroid is consistent and less than 8%. Interestingly, the error does not depend on the shape of the object and does not change across different population. However, the variance increases with the number of robots. This error is expected as the angular sensor errors are accumulated due to coordinate transform of the position.

Our tree-based algorithm for the centroid estimation is continually updated using current sensed values. This uses a great deal of communication resources, but provides the

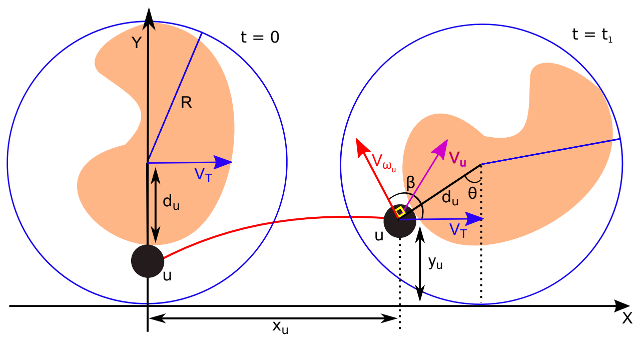


Fig. 6: Parameters for the trochoid. The trochoid is modeled by a point moving relative to a circle rolling on the ground without slipping. For the purposes of our algorithm, we simplify the object to be inside a rolling circle of radius R.

algorithm with robustness with regards to communication failure, population changes, and configuration changes. The algorithm is self-stabilizing and "forgets" previous states of the network, allowing errors to wash out after $O(2diam(G))$ rounds. The self-stabilizing property of the algorithm is shown in Fig. 4b. Starting with 5 robots, we add and remove robots during the experiment. The robots consistently update their estimate and track the changes in network population.

Consensus-based algorithm have a smaller per-round communication cost, $O(1)$, whereas our tree-based algorithm has a communications complexity of $O(n)$. However, the time for consensus to converge depends on the specific network, and the final error bounds that are desired. For example, to achieve an error less than 5% in a path network of diameter 3, a pairwise gossip consensus algorithm would require 10 rounds of computation, producing a 3.1% error[18]. Our tree-based centroid estimation completes in $2depth(T_i)$, where $depth(T_i)$ = the depth of tree T_i, which is bounded by $diam(G)$. As the depth of tree T_i increases, delay for the information to travel from the leaves to the root increases linearly. In our experiments, this delay is apparent for robots at the edge of the communication graph experience greater error due to the comparatively old data from far away robots. Our results show that despite this drawback, the estimate of the centroid is still practical technique on our real robot experiments involving up to 9 robots and a communication tree with a maximum depth of five.

IV. Distributed Motion Controllers

Our goal is to move the robot between two guide robots [2] with different configurations, which requires translation, rotation, combine motion of two. We have created four distributed primitive motion controllers for transport of an object: rotation about a pivot robot, rotation around the estimated centroid, translation towards a guide robot, and a combined controller with both rotation and translation that moves and rotates the object simultaneously. These four controllers are the primitive movements that will eventually lead to more complicated multi-robot manipulation of an object in a distributed manner. We have verified our controllers in real-world experiments.

Pivot Rotation The object can be rotated about a special pivot manipulation robot. Once a pivot robot has been selected, other manipulator robots align their heading perpendicularly to the location of the pivot robot and move with a speed proportional to their distance from the pivot robot (See Fig. 5a). This creates circular trajectories centered on the pivot with tangential velocity $V_u = \omega d_u$ for robot u, where ω is the desired angular velocity of the object and d_u is the distance of robot u from the pivot.

Each robot learns its vector to the pivot robot in their own local coordinate frame in a process similar to the centroid estimation. The pivot robot builds a broadcast tree rooted on itself (See Fig. 3). This broadcast tree relays the vector to the pivot down the kinematic chain. As the broadcast message propagates from parent to child, each child updates its vector to the root by using the parent's position in its coordinate frame to transform the parent's vector to the pivot into its coordinate frame.

Reading the subtree's pivot position, a robot can convert the position to its local coordinate frame. This is done by a series of transformations:

$$^{u}P = {}^{u}_{v}M \; {}^{v}Pu \tag{3}$$

where ^{u}P is the location of the pivot robot, P, in robot u's reference frame. v is u's parent in the tree rooted at pivot robot.

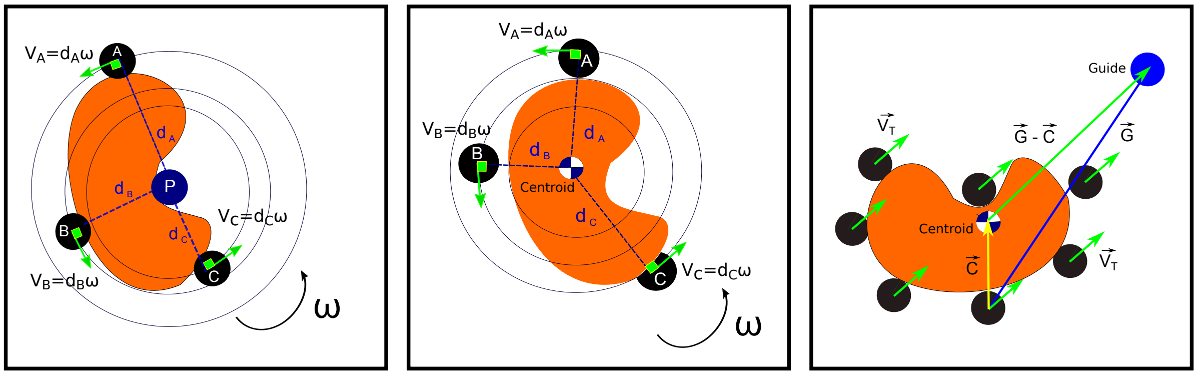
Centroid Rotation The object can be rotated about the estimated centroid of the object for a rotation in-place (See Fig. 5b). Using our centroid estimation algorithm, each manipulator robot can estimates the location of the centroid. We then use the same algorithm as the pivot rotation controller, the robots align themselves perpendicularly to the vector towards the centroid and command a tangential velocity proportional to their distance.

Translation The object can be translated towards a guide robot situated outside the object (See Fig. 5c). Each manipulator robot computes its vector to the guide robot, $\vec{G}$, using (3), the same algorithm for the pivot controller. Using their centroid estimate, $\vec{C}$, robots can find the desired vector of the object to the guide robot: $\vec{V_T} = \vec{G} - \vec{C}$. Then, the robot aligns itself parallel to this vector and commands the velocity $|\vec{V_T}|$ to transport the object.

Combined Controller The object can be rotated and translated towards a guide at the same time (See Fig. 9). Given a desired rotational (ω) and translation velocity (V_T) for the object, it moved and rotated along a straight path, but each robot moves along a trochoid path.

Fig. 6 shows the geometry for the combined controller for robot u. It generates the velocity $\vec{V_u}$, which is the vector sum of $\vec{V_{\omega_u}}$, where $|\vec{V_{\omega_u}}| = d_u\omega$ for rotational velocity, and $\vec{V_T}$ for the translational velocity. Assigning $|\vec{V_{\omega_u}}| = |\vec{V_T}|$ and regarding Fig. 6, the x and y components of $\vec{V_u}$ with angle β are derived as follows:

$$\vec{V_{x_u}} = |\vec{V_{\omega_u}}|\cos\beta + |\vec{V_T}| = d_u\omega\cos\beta + \vec{V_T}$$
$$\vec{V_{y_u}} = |\vec{V_{\omega_u}}|\sin\beta = d_u\omega\sin\beta$$



(a) Pivot Rotation **(b)** Centroid Rotation **(c)** Translation

Fig. 5: The three basic motion controllers. **(a)** Rotation about a pivot - Robots align themselves perpendicular to a pivot robot and rotate the object about the pivot. **(b)** Rotation about the centroid - Robots estimate the position of the objects centroid, align themselves perpendicular towards it, and rotate the object in place. **(c)** Translation - Robots translate the object along the vector from the estimated centroid to a guide robot.

Therefore, the x_u and y_u position of robot u is computed as:

$$\begin{aligned}
x_u &= \int_0^t \vec{V_{x_u}}\,\mathrm{d}t + X_0 = \int_0^t \vec{V_{x_u}}\,\mathrm{d}t + 0 \\
&= \int_0^t d_u\omega\cos(90+(90-\theta))\,\mathrm{d}t + \int_0^t R\omega\,\mathrm{d}t \\
&= -d_u\sin\theta + R\theta \\
y_u &= \int_0^t \vec{V_{y_u}}\,\mathrm{d}t + Y_0 \\
&= \int_0^t d_u\omega\sin\beta\,\mathrm{d}t + (R-d_u) \\
&= -d_u\cos\theta + d_u + R - d_u = -d_u\cos\theta + R
\end{aligned}$$

This derivation gives us:

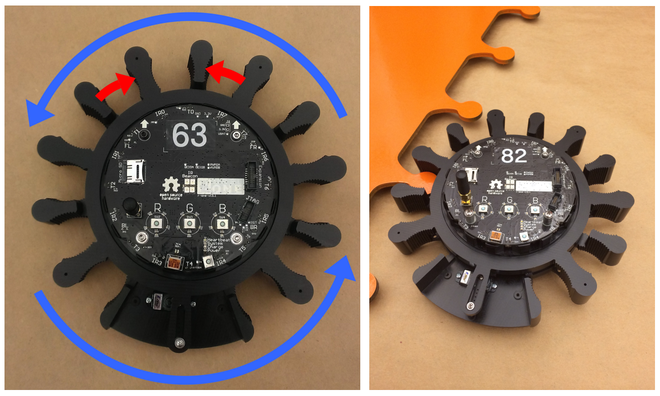
$$x_u = R\theta - d_u\sin\theta \qquad y_u = R - d_u\cos\theta \tag{4}$$

which is the equation for the position of a point d_u distance from the center of rotation. This point moves on a trochoid path. $\square$

Therefore each robot can calculate its desired velocity $\vec{V_u}$ using the centroid estimate, $\vec{C}$, and the guide position, $\vec{G}$, from Fig. 5c. With each manipulator moving on a trochoid path, the object moves along a straight path while rotating.

V. Hardware Experiments

We used the r-one robot platform [1] for our hardware experiments. This is the same robot platform we used in our previous work on distributed path planning [2]. We tested our algorithms using groups of 2-9 robots. The r-one robots use an infrared communication system communicate with their neighbors in synchronous rounds of 1500 milliseconds (see Section II). The communication system also measures the relative pose to each neighboring robot. However, the range measurement component of neighobr pose was still under development, so we used the AprilTag system [19] to determine both ground-truth pose of each robot for data collection, and to broadcast inter-robot ranges to each robot.



(a) Gripper **(b)** Gripping an object

Fig. 7: **(a)** A single r-one robot with the gripper attachment. Alternate paddles can move to oppose each other to grasp objects. The red arrows show the gripping motion of the paddles. The entire gripper is free to rotate around the robot, as shown by the blue arrows. **(b)** The gripper can grab any object equipped with the appropriate "handles". Because the griper is free to rotate around the robot, the robot can exert a force on the object, but not a torque.

A. Gripper

The r-one gripper allows the robot to grasp onto objects with appropriately sized "handles", shown in Fig. 7b [1]. The gripper can grasp from any direction, and is free to rotate around the robot, shown in Fig. 7a. This unconstrained rotation allows us to simplify our model for object transportation; an individual robot is a force source, and cannot exert a torque. The current revision of the gripper has considerable friction around the free pivot point, and this puts some limitations on our controllers, which we will discuss in our results section. Also, the free-rotating assembly has a 330°rotation limit to prevent winding of the cord that attaches the gripper control board to the r-one. The distance between the attachment point of the gripper on the object to the robot's point of contact with the ground can create a torque between the robots and the object, which tends to perturb the robot's heading as they are controlling the object.

B. Motion Controllers

The tests of the motion controllers were started with the robots positioned and attached along the perimeter of a test object. In order for the robots to overcome the frictional forces and rotational constraints introduced by the grippers, the robots were allowed to align themselves towards their desired vector at the beginning of the experiment so they would not get stuck against the limit stop of the gripper.

We tested each motion controller for 5-7 robots, 12 trials each, for 96 total trials. We compared the desired values of the object's translational velocity TV and rotational velocity RV against our measured values. Fig. 8 shows the results for all controllers. In general, the performance was quite good, with all the controllers producing well-controlled motion. In all the controllers, wheel slippage and friction were the dominant sources of error, making it difficult to specify the exact speed, however, all the speeds produced had low variance . Detailed discussion follows.

Pivot Rotation The pivot controller data is shown in Fig. 8a. The translational velocity at the center of rotation (the pivot robot) was nearly zero, while the rotational velocity shows the value for the object. The errors in the expected to actual values come from conflicting forces other robots exerted on the object, the force that the object then exerted on the robot, and the constant adjustments that the robots needed to make to continue to apply force along the desired vector. The TV of the center of rotation shown in Fig. 8a is actually a tangential velocity of the pivot robot as it is pulled in a circular path by the other robots rotating the object around it.

Centroid Rotation Fig. 8b shows 12 trials of centroid rotation. The object showed a mean RV slightly under the desired value of 5.84 degrees per second and a TV close to 0. Like in the pivot, the other robot's forces limit the robot's potential to achieve the desired velocities. The TV of the centroid is also a tangential velocity of a circular path. This is due to the robots aligning themselves towards the approximated centroid, not the actual, making the center of rotation slightly skewed.

It should be noted that the trials with 6 manipulator robots had noticeably better performance in the rotational motion controllers than either the 5 or 7 robot trials. This is due to the distribution of the robots around the object. Due to the shape of that object, the more symmetrical distribution of robots about the perimeter allowed for more accurate estimation of the centroid and smoother rotation.

Translation Fig. 8c shows the results of experiments with the translational controller. Our experimental data shows the small difference between the desired TV of the object to the actual, and the small amount of RV the object experiences in the real experiment. Just as in the rotation about the pivot, other forces limit the robot's potential to achieve the desired velocities.

In the case of this controller, more robots added to the task of manipulation improves the motion of the object. More robots applying force to the object allows for smoother, straighter movement.

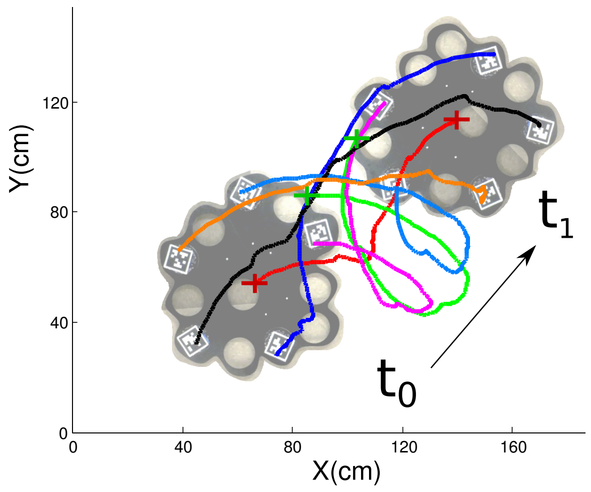


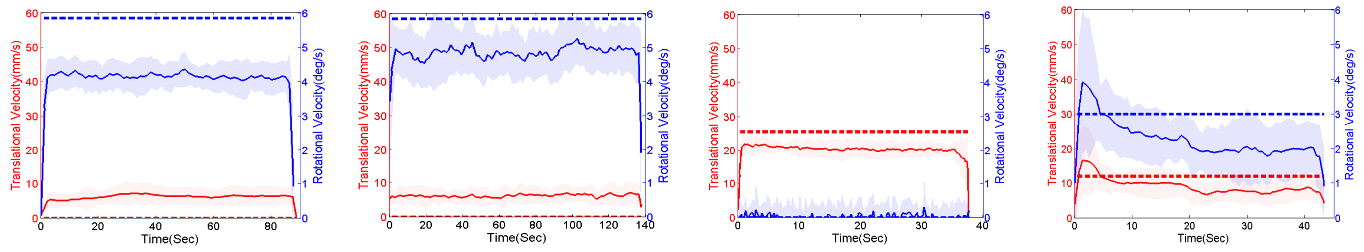
Fig. 9: The trajectory of the object and manipulator robots when moving using the compound controller from initial time t_0 to t_1. Each robot follows a trochoid path. The centroid moves along a (mostly) straight line, and is represented by the red point and path.

Combined Motion For the tests of the compound trochoid motion controller, we placed the r-one robots inside holes of an object (Fig. 9) instead of using grippers as the frictional and motion constraints caused by the gripper attachments are too great to accurately test the capabilities of this controller. This controller was the most sensitive to the friction and inter-robot forces. This is because out of the 4 controllers, this is the only one that has the robots moving relative to each other in the ideal case. The overshoot at the beginning of the trials is most likely caused by the difficulty in breaking static friction, but then the velocities stabilized close to the desired values.

VI. Discussion and Limitations

The first source of error comes from the inaccuracy of local pose estimation between neighboring robots. For centroid estimation, the limited angular resolution of our sensors is compounded as the tree's depth increases, and stacked angular pose estimations cause error to propagate up the tree. A poor angular measurement near the root of the tree can cause estimated data from all children to be off by a wide margin. However, the centroid error is still small, so this is not a major impact.

The largest source of error in our experiments is the friction between the robots and the grippers, which were compounded by the exogenous forces exerted on the robots by others moving the object. In the process of conducting this work, we have upgraded many parts to low-friction teflon versions. This helps, but the core limitation is that the robots cannot sense the forces that are being applied to them. This is because we have only studied *kinematic* controllers in this work, handling external forces will require *dynamic* controllers. This is an exciting area for future work, and design of a force-sensing gripper is already underway.



(a) Pivot rotation **(b)** Centroid rotation **(c)** Translation **(d)** Combined(Trochoid)

Fig. 8: The comparison of performance of controllers in hardware experiments. In all these plots, the measured translational and rotational velocities of the object are shown in red and blue. The mean is drawn as a solid line and the variance is shaded. The desired velocities are shown with dotted lines. **(a)** Rotation around the pivot. In this plot, the TV shown is for the pivot robot. **(b)** Rotation around the estimated centroid. **(c)** Translation toward the guide robot. **(d)** Combined motion. Each robot follows a trochoid path.

VII. Conclusion

We introduced four distributed motion controllers for multi-robot systems to collectively manipulate a large object. In addition, we also developed a novel centroid estimation algorithm to provide an accurate and stable way for the robots to gain a reference point on the object without explicit knowledge of the geometry of the object. With the controllers and centroid calculation as building blocks, multiple robots can rotate and translate an object in a variety of different trajectories, allowing easier traversal around obstacles.

Our goal is to integrate these controllers with our previous work in path planning [2] to create a complete multi-robot transport solution. Using planning robots distributed across the environment, the motion controllers from this paper can follow the planned path to maneuver an object from the start to the goal positions. There are many open problems with this approach, such as coordination between guide and gripper robots, avoiding collisions during transport, and deciding the proper combinations of the controllers to use. We look forward to these topics in future work.

ACKNOWLEDGMENT

The authors would like to thank Kai Holnes and Sujay Tadwalkar for their numerous efforts in collecting data.

References

[1] J. McLurkin, A. McMullen, N. Robbins, G. Habibi, A. Becker, A. Chou, H. Li, M. John, N. Okeke, J. Rykowski, S. Kim, W. Xie, T. Vaughn, Y. Zhou, J. Shen, N. Chen, Q. Kaseman, L. Langford, J. Hunt, A. Boone, and K. Koch, "A robot system design for low-cost multi-robot manipulation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*. IEEE, 2014, pp. 912–918. [Online]. Available: http://dx.doi.org/10.1109/IROS.2014.6942668

[2] G. Habibi, W. Xie, M. Jellins, and J. McLurkin, "Distributed Path Planning for Collective Transport Using Homogeneous Multi-Robot Systems," *Proc. of the International Symposium on Distributed Autonomous Robotics Systems*, 2014.

[3] Y. Le, H. Kojima, and K. Matsuda, "Cooperative Obstacle-Avoidance Pushing Transportation of a Planar Object with One Leader and Two Follower Mobile Robots," *Journal of Robotics and Mechatronics*, vol. 17, pp. 77–88, 2004.

[4] G. A. S. Pereira, V. Kumar, and M. F. M. Campos, "Decentralized algorithms for multirobot manipulation via caging," *International Journal of Robotics Research*, vol. 23, pp. 783–795, 2002.

[5] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-Robot Manipulation via Caging in Environments with Obstacles," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1471–1476, May 2008.

[6] Q. Chen and J. Y. S. Luh, "Coordination and control of a group of small mobile robots," pp. 2315–2320 vol.3, May 1994.

[7] R. Groß, E. Tuci, M. Dorigo, M. Bonani, and F. Mondada, "Object Transport by Modular Robots that Self-assemble," no. May, pp. 2558–2564, 2006.

[8] R. Gross and M. Dorigo, "Towards group transport by swarms of robots," *Int. J. Bio-Inspired Comput.*, vol. 1, no. 1/2, pp. 1–13, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1504/IJBIC.2009.022770

[9] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal, "Collective transport of complex objects by simple robots: Theory and experiments," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 47–54. [Online]. Available: http://dl.acm.org/citation.cfm?id=2484920.2484932

[10] Z. Wang and M. Schwager, "Multi-Robot Manipulation without Communication," *Proc. of the International Symposium on Distributed Autonomous Robotic Systems*, 2014.

[11] S. Berman, Q. Lindsey, M. S. Sakar, V. Kumar, and S. C. Pratt, "Experimental Study and Modeling of Group Retrieval in Ants as an Approach to Collective Transport in Swarm Robotic Systems," *Proc. of the IEEE*, vol. 99, pp. 1470–1481, 2011.

[12] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325 – 349, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870503000738

[13] J. McLurkin, "Analysis and implementation of distributed algorithms for multi-robot systems," Ph.D. dissertation, MIT ,USA, 2008.

[14] M. Franceschelli and A. Gasparri, "Decentralized centroid estimation for multi-agent systems in absence of any common reference frame," in *American Control Conference, 2009. ACC '09.*, June 2009, pp. 512–517.

[15] M. A. Batalin and G. S. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *in Proc. of 6th International Symposium on Distributed Autonomous Robotic Systems*, 2002, pp. 373–382.

[16] G. Habibi, Z. Kingston, Z. Wang, M. Schwager, and J. McLurkin, "Pipelined consensus for global state estimation in multi-agent systems," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS '15. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[17] J. McLurkin and D. Yamins, "Dynamic task assignment in robot swarms," in *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[18] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[19] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.