Predicting Center of Mass by Iterative Pushing for Object Transportation and Manipulation

Steven M. Hyland, Jing Xiao, and Cagdas D. Onal

***Abstract*—Robotic manipulation tasks rely on a plethora of environmental and payload information. One critical piece of information for accurate manipulation is the center of mass (CoM) of the object, which is essential for estimating the dynamic response of the system and determining the payload placement. Traditionally, the CoM of a payload is provided prior to manipulation. In order to create a more robust and comprehensive system, this information should be collected by the robotic agent before or during the task run time. This paper presents a method for approximating the CoM of a planar object using a small-scale mobile robot to inform manipulation tasks. On average, our system is able to converge on a CoM estimate in under 30 seconds in simulation and 20 seconds in experiment, with a relative error of 4.95% and 5.46%, respectively.**

I. Introduction

Robotic manipulation focuses on moving an object, henceforth referred to as the 'payload', from an initial position to a desired final pose. Traditional implementations rely on knowledge about the environment and payload, such as localization information and payload inertial parameters. Robotic manipulation relies on this information for two purposes: 1) to understand how the grasped object will behave while in motion, 2) to plan a placement strategy that does not result in instability. In this study, we focus on reducing the reliance on *a priori* knowledge of the inertial parameters, specifically, the center of mass (CoM) of rigid payloads through estimation during the manipulation task.

Inertial parameters are a function of an object's volume, volumetric distribution, and density [10]. Of these parameters, CoM information is particularly useful for determining an object's interactions with the environment and robotic agents. While humans are skilled at intuitively determining the CoM of objects, either by a shift of the hand or by visual inspection, the same cannot be said for automated systems. This point is further reinforced when the object in question has a non-uniform mass distribution, for instance, a loaded cardboard shipping box. In this case, visual methods become unfeasible, and some other approach must be taken.

This paper presents a method of estimating the center of mass (CoM) of arbitrary objects via iterative pushing, both in simulation and with a physical robotic experiment. Our method converges to an accurate CoM prediction rapidly during the manipulation task itself. Furthermore, the algorithm requires very little prior knowledge of the payload, and is applicable for a majority of rigid objects. The robotic platform used to test our algorithm is based off the *Delta-rho* system used for collective manipulation in [2], consisting of a mobile robot for pushing the object.

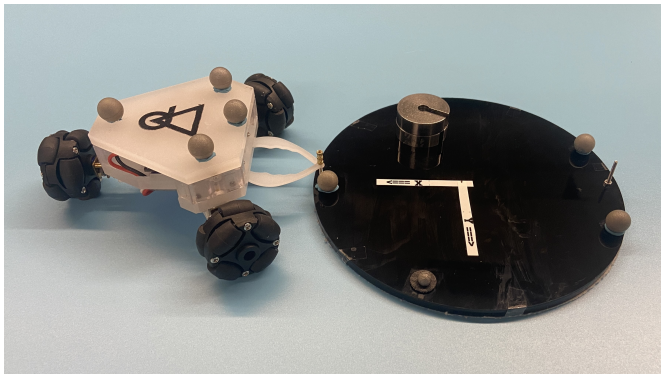

Fig. 1. Physical System Setup. OptiTrack tracking markers are placed atop both the robot and the payload to capture their position and orientation in real time. A Mass was shifted to different positions on the payload to change the CoM location. Markers were set at non-uniform intervals to discretize the rotation of the object.

Our group's previous work [2], [1], [5], and [6] focus on 2-dimensional collective manipulation using a decentralized robot swarm, consisting of individual small mobile robots. For these studies, however, the CoM location is either assumed to be at the centroid (geometric center) of the object, or must be provided *a priori* to the agents. The CoM is utilized to determine the desired final pose of the object and to inform the controller which handles the swarm's motion plans.

There are a number of approaches in the literature to solve this problem autonomously, [3], [4], [10], [11] all estimate the CoM using a virtual robotic manipulator and a complex dynamical model of both dynamic pushing and quasi-static pushing. [7] fortifies this approach by developing a deep learning *Push-Net* model for iteratively learning the CoM. The authors investigate their method on a Kinova MICO arm, utilizing an RGBD camera for visual feedback. A unique approach to the problem is explored in [8], where only tactile feedback is used to determine a pushing scheme. This force response allows for tetherless predictions, avoiding the limitations implicated by vision or motion capture. However, the detectable contact orientation is limited by the tactile sensors.

Our earlier work in [12] and [13] researches an alternative approach, bypassing pushing procedures. We apply a reinforcement learning algorithm to improve the estimated CoM over time while grasping the payload and leveraging the environment. We implement a virtual and physical robot arm to examine two methods: 1) balancing a payload at the edge of a flat surface until tipping occurs, 2) lifting a payload on

multiple sides, gathering torque data to inform the algorithm.

The previously discussed studies require complex artificial intelligence frameworks, dynamic modeling, or both to determine the CoM. Conversely, the approach in this paper avoids these computational costs and prior knowledge by leveraging environmental friction properties. By employing a modified binary search algorithm [14], in conjunction with voting theorem (VT) [9], the line of action through the CoM can be estimated from any point on the edge of the payload. Through this, a dynamic model of the system becomes redundant; these implications are discussed further in section II. While our proposed method employs off-board vision-based control, it can be supplemented or entirely replaced with a force or torque sensor for tetherless CoM estimation.

II. Problem Definition and Methodology

Given an arbitrary payload with an unknown mass distribution, this robotic system estimates the 2-dimensional coordinates of the CoM on the object's surface with respect to its centroid. All computation is performed remotely, allowing the robotic agent to be as simple as possible. The algorithm inputs are the robot and payload pose, and the attachment point with respect to the payload's centroid at every time step. Figure 2 illustrates the algorithm in action.

Some minimal assumptions are made prior to executing the search. This work only estimates the 2-dimensional (x, y) CoM location, and assumes the robot is able to maneuver the payload. While the robot used in this research is lightweight and can therefore only manipulate objects with similar mass, the overarching approach is applicable to a robot with greater manipulation capacity. This research also assumes the robot begins pre-attached to the object by a singular rotary joint at a fixed location, as opposed to a pinch-grasp or prehensile grasping. A final assumption is that the interface between the payload and the driving surface exhibits isotropic friction. In other words, this version of the algorithm assumes the center of mass and center of friction are coincident. While this algorithm can be expanded to include these scenarios, this case is beyond the scope of this work.

A. CoM Search Algorithm

Before beginning the search, some user-defined parameters must first be selected. These parameters are t_{check}, θ_{max}, and θ_{min}. Establishing appropriate values for these parameters is non-trivial: the specific values and reasoning is elaborated further in Section V.

The algorithm begins by determining the initial bounds of the binary search. These bounds describe the region within which the CoM may lie. To preserve the simplicity of the algorithm, the positive bound, $bound_{(+)}$ and negative bound, $bound_{(-)}$ are initialized as $\pm 90°$ perpendicular to the agent's attachment point to the payload, r, respectively. In practical terms, the bounds are initialized tangent to the payload at r. Next, the robot's initial push direction must be calculated. This push vector, $heading$, throughout the procedure is calculated by bisecting the two bounds. This bisection is rapidly calculated using Algorithm 2. In the initial case, where the bisection of the bounds separated by $\pm 180°$ results in a singularity, the tie is broken by initializing $heading$ as $-r$. Thus, the initial push direction is set to always point towards the payload's centroid. The CoM search algorithm is extended from a traditional binary search, in which the search interval is regularly halved. The search interval bounds are periodically updated as the robot takes measurements of the payload's rotation.

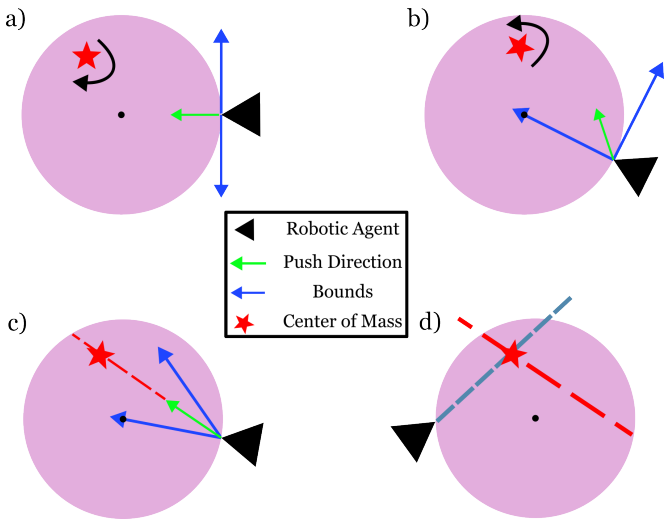

Fig. 2. Illustration of the CoM estimation algorithm process. Payload depicted by pink circle. a) The initial bounds are tangent to the object at the robot's attachment point. Initial push direction bisects these bounds towards the payload centroid (black dot). This causes clockwise rotation around CoM. b) Positive bound is updated towards the previous push direction. Again, new direction is set bisecting the bounds. This causes counter-clockwise rotation. c) Next, the negative bound is updated. New push direction is along the CoM line of action, no rotation is measured. d) Line of action recorded. Repeat process from another attachment point to acquire another line. Intersection is found for the final CoM estimate.

The robot measures the payload's rotation, θ_t, every time step, until one of two events occur: 1) θ_t exceeds the set threshold, θ_{max}, or 2) θ_t is below the *measurable* rotation threshold, θ_{min}. In the first case, the bounds are updated according to Algorithm 2, and a new direction is decided which satisfies the condition of bisecting the bounds. The second case acts as the stop criteria for the simulation, however, a second condition must first be met. That second condition compares the time, t, since the previous measurement, to the measurable rotation time threshold, t_{check}. This condition ensures that the payload exhibits minimal rotation for a certain span of time. Consequently, if the stop condition is met, the search ends and the final push direction is captured. This final push direction describes the 'line of action' (LoA) of the CoM; the robot estimates that the CoM lies somewhere along this line.

The entire process above corresponds to a single attachment point r. This process is repeated with the robot attached to a separate r. The greater the number of iterations that are performed from different values of r, the higher the accuracy of the overall estimate. This study only performs two iterations, from two attachment points, to examine a worst-case estimate. With these two CoM lines, their intersection

is calculated with respect to the object, and the coordinates of the CoM prediction are given.

The selection of distinct attachment points is not trivial; there exists certain singularities. Consider the case of the CoM coinciding the centroid of the circular payload. If the attachment points are at 0° and 180° from the $+x$ axis, the estimated lines of action may be colinear and not intersect, making an estimate impossible to generate. Thus, the two attachment points were selected 90° apart to minimize the possibility of being almost parallel due to uncertainty. Alternatively, we can select three or more attachment points and guarantee non-colinearity. A future study will examine these implication in further detail.

In this section and beyond, one 'iteration' refers to an agent pushing the payload from a single particular attachment point; two iterations are required at minimum to make an estimate.

III. Simulation

The prediction algorithm is simulated in MATLAB. The robotic agent is modeled as a point force, and uses acceleration control to more closely model the behavior of the physical experiment. That is, at every time step, the simulation calculates a force that the agent applies to the payload, which is then integrated using MATLAB's *ode45* ordinary differential equation solver to model the system dynamics. The direction of the applied force and the acceleration of the agent is dependent on each step's bound update results, delineated in Algorithms 1 and 2.

In order to model the behavior of the payload with respect to the agent's pushes accurately, a dynamic and contact model is used. The dynamic model considers isotropic friction acting in the primary (X, Y) plane, as well as rotational friction about the Z axis. To satisfy the point force constraint, the agent's velocity is set equal to the velocity of the attachment point, r, on the payload. This culminates in the agent following its *approximate* desired trajectory, without neglecting the influence of the payload's inertia. The CoM location is adjustable to anywhere on the face of the object by providing coordinates with respect to the object centroid. As explained below, this is a realistic model of the real world interactions.

IV. Physical System

The physical robot platform used for this research, shown in Figure 1, was created and developed in [2], where it was used for 2-dimensional collective manipulation.

In order to change the location of the CoM experimentally, a $100\ g$ weight was placed on top of the payload in specific locations. A critical difference to simulation is that the CoM is not necessarily at the coordinates where the weights are placed. Instead, an averaging function is used to calculate its *actual* position, seen in Equation (1). In this type of dual-mass system, the actual CoM lies at a point between the two masses - in this case, between the payload's centroid, p_1, and the placement of the weights, p_2.

Algorithm 1 Center of Mass *Line of Action* Search

procedure CoM($r, \theta_t, \theta_{min}, \theta_{max}, d_{init}, t_{check}$)
  $bound_{(\pm)} \leftarrow \pm \perp$ *to* r
  $heading \leftarrow d_{init}$
  while $\theta_t > \theta_{min}$ **do**
    $t \leftarrow t + 1$
    robot.$move(heading)$
    $\theta_t \leftarrow$ robot.$measure()$
    $t, heading \leftarrow$ NewBounds($\theta_t, bound_{(\pm)}, heading$)
    if $t > t_{check}$ **then**
      if $\theta_t < \theta_{min}$ **then**
        break
      else
        $t \leftarrow 0$
  CoM(r) $\leftarrow heading$

Algorithm 2 Update Bounds Based on Payload Rotation

procedure NewBounds($\theta_t, \theta_{max}, bound_{(\pm)}, heading$)
  if $\theta_t < \theta_{max}$ **then**
    continue
  else
    if $\theta_t > 0$ **then**
      $bound_{(-)} \leftarrow heading$
    else
      $bound_{(+)} \leftarrow heading$
    $bisect \leftarrow bound_{(+)} + bound_{(-)}$
    $heading \leftarrow bisect_{unit}$
    $t \leftarrow 0$

$$CoM = \frac{m_1 \cdot p_1 + m_2 \cdot p_2}{m_1 + m_2} \tag{1}$$

$$e_r = atan2\Big(n_o W, n_r W\Big) \tag{2}$$

$$u_r = \begin{cases} e_r & \text{if } e_r \leq \pi \\ A \cdot (B^{\,e_r + C}) & \text{if } e_r > \pi \end{cases} \tag{3}$$

For the experiment, the same general CoM search process was performed, with certain modifications. The principal difference between the simulation and the physical experiment is how the robotic agents are modeled. In simulation, the agents were modeled as point forces along the perimeter of the payload. This indicates that the simulated agent has no dynamics - it has no rotational component, nor does it consider any wheel traction with the driving surface. The physical robot does indeed have both considerations to contend with. The robot rotation is accounted for by implementing an attitude correction term, e_r, which leverages null space control to remain normal to the payload at all times, seen in Equation (2). This term minimizes the error between the robot's heading, $n_r W$, and the payload's heading, $n_o W$, while rotating about its own grasp point. The strength of this attitude correction term is calculated through a piecewise exponential controller, described in equation (3). π represents

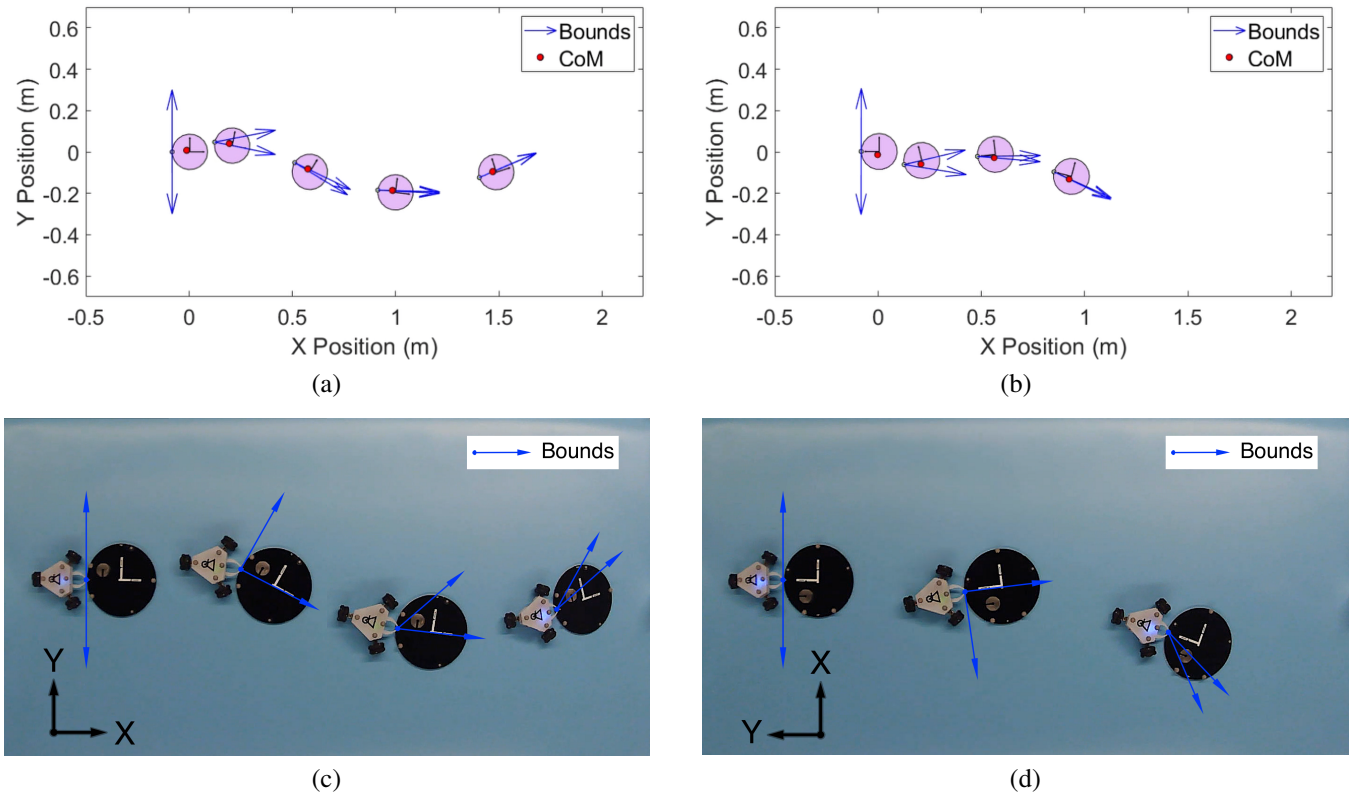

Fig. 3. Estimates of CoM of test position 4: (-15.12,7.56). Payload is rotated instead of camera view for consistency. (Top) Simulation of the CoM estimate algorithm. The circular payload in pink is pushed by the agent. The bounds (blue arrows) capture the region within which the CoM may lie, updating after each measurement. a) The agent pushes the object from attachment point 1, the rightmost vertex of the circle until a CoM line of action is determined. b) The same process is repeated from attachment point 2, the topmost vertex of the circle. (Bottom) Experiment estimate. c) The robot pushed from leftmost point. d) The robot pushes from topmost point.

the permissible lower bound for correction amplification: if the raw error is below this number, it is passed to the robot without modification. This allows for small attitude errors, and prevents over-correction. The remaining constants, A, B, C, are discussed in further detail in section V-C.

$$J_r = \sum_{w=1}^{3} \begin{bmatrix} -sin(\alpha_w) \\ cos(\alpha_w) \\ \rho_{wx}cos(\alpha_w) - \rho_{wy}sin(\alpha_w) \end{bmatrix} \tag{4}$$

The null space control comes from equation (4), representing the robot Jacobian from [6]. This allows for robot rotation about its end effector to maintain a certain heading while also avoiding collision with the payload. Here, w refers to each of the three wheels, α refers to the angle between the wheels, and ρ_w refers to the x and y positions of each wheel.

V. Results and Discussion

For both the MATLAB simulation and the experiment, a circular payload of radius $86mm$ and mass $180g$ was tested. Five different CoM positions were selected, with five trials performed for each position. For the simulation, only a single trial was conducted per position due to repeatability. A control experiment was conducted with the CoM positioned at the payload centroid. The % error is defined as the Euclidean distance between the estimate and the actual CoM, as a fraction of the payload's diameter. The five CoM positions and the simulation and experimental results are shown in Table I and Table II. The coordinates in the tables have been rounded to one decimal for legibility.

For the sake of brevity, only the **fourth** CoM position, $(-15.12, 7.56)$, is visualized by Figure 3.

A. Parameters

Due to the accurate and repeatable nature of simulations, we have the capacity to narrow the parameters to achieve optimal results for the CoM search. However, to maintain a fair comparison between the physical experiment, the parameters were kept consistent across all trials and between the simulation and experiment. The three parameters were set to: $t_{check} = 3sec$, $\theta_{min} = 3°$, and $\theta_{max} = 10°$.

The parameters t_{check} and θ_{min} define the stop condition once minimal payload rotation is measured for some time. Decreasing θ_{min} or increasing t_{check} amplifies the stiffness of the system, and may result in a more accurate estimate at the cost of longer run time, with the opposite case also holding true. Conversely, θ_{max} defines the *bound update* condition. In simulation, this parameter is valid as long as $\theta_{max} > \theta_{min}$ due to the oracle knowledge of the payload rotation. However in the experiment, wheel slippage and motion capture occlusion demands a larger difference between the two parameters. This parameter has a direct relationship

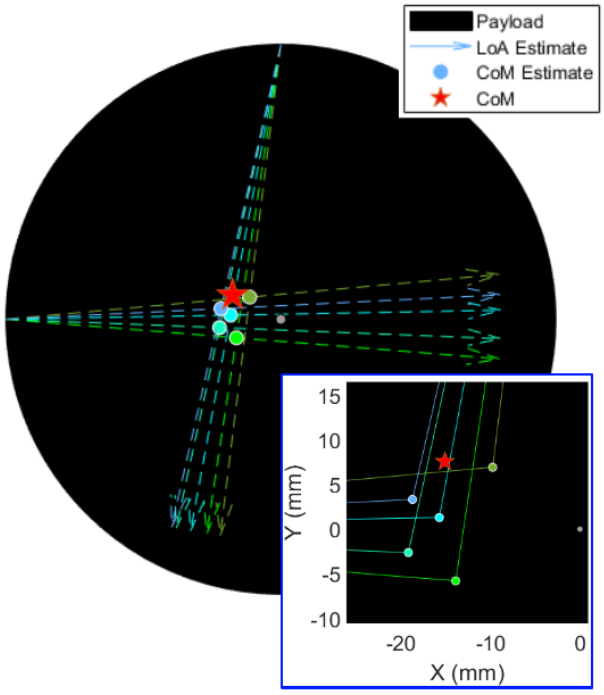

Fig. 4. Experimental CoM estimates for position 4: (-15.12,7.56). Each pair of colored lines represents a single pair of line of action (LoA) estimates. Two attachment points at 180° and 90° from positive x-axis.

with run time. We found that $\theta_{max} = 10°$ results in accurate estimates, while avoiding large sacrifices to total execution time.

B. Simulation Results

MATLAB's *ode45* solver allowed for the dynamics to be accurately modeled across the simulation. "Measurements" taken by the robot are innately known, and provided at every time step. Over all five CoM positions, the system converged to a CoM line of action (LoA) prediction in an average of 13.8 seconds, and a total average run time of 27.6 seconds. Depending on the position of the CoM, a different moment will be generated at the agent attachment point, hence the discrepancy between the completion time and rotation magnitude between tested positions. Figure 3 shows the agent performing the search from the two specified attachment points above. The robot moves from left to right in all figures. Table I displays the predicted CoM vs actual CoM location, as well as the relative error and run time of the simulated system.

As mentioned previously, only a single trial was conducted for each CoM position in simulation.

C. Physical Experiment Results

The physical agent is a small, $160mm$ diameter and $250g$ holonomic mobile robot housing the *ATmega1284p* micro controller as the processor for its custom circuit board. The three-wheeled holonomic drive allows for the robot to apply any desired 2-dimensional force to its payload, regardless of its orientation. The robot is attached to the payload with a standoff, connecting a particular edge point to the robot's end effector, which is a non-actuated piece of acrylic. The robot's end effector is permitted to swivel about this point. An Opti-Track Motion Capture System was used to collect the robot and payload poses, and the robot's rotation measurements. The robot and payload are shown in Figure 1. The payload was created from a laser cut $1/4\,in$ acrylic sheet, constructed to match the simulation. The payload had a mass of $165\ g$, and the CoM was modified by shifting a $100\ g$ weight to specific locations for testing. As described in Section IV, the weight was placed such that the actual CoM locations were consistent between the simulation and experiment. The same procedure as the simulation was conducted on the physical hardware, Figure 3 correspond to the two push iterations for the fourth selected CoM position.

This robot was not a point force as modeled in simulation, consequently, its attitude was corrected to avoid collisions with the payload. As explained in Section IV, this was performed by maintaining the robot normal to the payload's edges throughout the search. Since the robot has three degrees of freedom and two actively controlled push vector components, we used the remaining control authority to execute this task. The intensity of this correction term was determined through trial-and-error to select appropriate parameters to the exponential function in equation (3). The parameters that produced a desirable correction response were: $A = 1.5$, $B = 1.1$, and $C = 1.5$, for a final exponential formula in equation (5), wherein the piecewise formulation did not amplify minor attitude errors.

$$u_r = \begin{cases} e_r & \text{if } e_r \leq 5 \\ 1.5 \cdot (1.1^{\,e_r + 26}) & \text{if } e_r > 5 \end{cases} \tag{5}$$

A CoM LoA was estimated on average in 9.6 seconds, for a total run time of 19.2 seconds. Figure 4 visualizes the five trials for the selected CoM position. The clustering of the estimates indicates relatively high precision, although the majority of the group was off from the actual CoM by a few millimeters. The attached video shows the CoM estimation process both in simulation and in experiment.

TABLE I
SIMULATION RESULTS

	Actual (mm)	Estimate (mm)	Error (mm)	Error (%)	Time (s)
1	(0,0)	(0,0)	0	0	6.0
2	(22.7,-15.1)	(15.0,-9.4)	9.59	5.57	39.9
3	(18.9,18.9)	(20.1,9.2)	9.81	5.71	36.9
4	(-15.1,7.6)	(-13.7,1.9)	5.85	3.40	37.5
5	(-11.3,-17.0)	(-3.0,-14.2)	8.78	5.11	17.5

TABLE II
EXPERIMENT RESULTS

	Actual (mm)	Estimate (mm)	Error (mm)	Error (%)	Time (s)
1	(0,0)	(0.4,2.7)	2.72	1.58	16.9
2	(22.7,-15.1)	(8.2,-9.3)	15.59	9.06	19.0
3	(18.9,18.9)	(15.9,11.9)	7.64	4.44	18.0
4	(-15.1,7.6)	(-15.3,0.6)	6.95	4.04	13.6
5	(-11.3,-17.0)	(-12.0,-9.6)	7.42	4.32	18.5

D. Discussion

The overall average accuracy and relative error for all positions and trials in simulation were, 8.51*mm* and 4.95%, and for experiment were, 9.40*mm* and 5.46%, respectively. As expected, the results from simulation were superior to the physical experiment, in accuracy and error. Naturally, the experiment has more variability than simulation, although as discussed previously, the largest culprit for the discrepancy between the simulation and experiment is the attitude correction of the robot. The attitude adjustments successfully prevented collisions, however, caused undue directional changes to the robot's push vectors.

The simulation had a longer overall execution time for each CoM position compared to the experiment. This is likely due to the force magnitude imposed by the agent onto the payload. The simulation approximates real-world dynamics, although the force output by the agent in the experiment is difficult to ascertain without the use of a force sensor. A nominal value of $0.4N$ was used as the maximum permissible force the agent applied. However, this longer run time is also characterized by a more accurate result.

Interestingly, certain CoM positions yielded more precise estimates. For example, CoM positions two and three, $(22.68, -15.12)$ and $(18.90, 18.90)$, respectively, resulted in some of the highest average errors of all the trials. These two positions are located in the two furthest quadrants from the robot's attachment point, r. This increased distance from r yields a larger lever arm, which in turn produces an increased moment. This larger moment causes a faster rotation and an earlier bound update by the robot, however the robot's response to change directions may not be quick enough to overcome the system's residual momentum. One solution to this problem is to bring the system to rest for a moment after every measurement. This way, there are no lingering dynamics to affect future pushes.

VI. Conclusion

This paper demonstrated the feasibility of finding the two-dimensional center of mass (CoM) location of arbitrary objects using a mobile robot platform. Furthermore, the robot platform used for this study was previously used for collective manipulation, demonstrating that this process can be quickly performed during run time or just before collective manipulation, and even other general manipulation tasks. For objects with unknown mass distribution, or that have an occluded non-obvious CoM, the information this method provides can be used for other purposes, not necessarily limited to robotic manipulation. These objects traditionally prove difficult to extract CoM information from due to their irregular characteristics.

The proposed approach was able to converge on a CoM prediction within 30 seconds. Small deviations in starting conditions and positions did not affect the accuracy of the prediction.

A future extension is to adapt the system for thetherless operation. That is, to remove the reliance on the motion capture system for recording object rotation. One potential avenue is to replace the end effector with a force sensor which provides force direction and magnitude. Another area of work is to test different approaches to determining the center of mass, for example, by pulling the object instead, or by using a learning framework to inform the robot over time. Finally, another possible extension of this work would be to test this proposed procedure during an actual collective manipulation task.

References

[1] Siamak G. Faal, Shadi T. Kalat, and Cagdas D. Onal. Decentralized obstacle avoidance in collective object manipulation. In *2017 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 133–138, Pasadena, CA, USA, 2017. IEEE.

[2] Siamak G. Faal, Shadi Tasdighi Kalat, and Cagdas D. Onal. Towards collective manipulation without inter-agent communication. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 275–280, Pisa Italy, April 2016. ACM.

[3] Ziyan Gao, Armagan Elibol, and Nak Young Chong. Estimating the Center of Mass of an Unknown Object for Nonprehensile Manipulation. In *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1755–1760, Guilin, Guangxi, China, August 2022. IEEE.

[4] Ziyan Gao, Armagan Elibol, and Nak Young Chong. Estimating the Center of Mass of an Unknown Object Via Dynamic Pushing. In *2022 IEEE International Conference on Automation Science and Engineering (CASE)*, page 4, Mexico City, Mexico, 2022. IEEE.

[5] Shadi T. Kalat, Siamak G. Faal, and Cagdas D. Onal. Scalable collective impedance control of an object via a decentralized force control method. In *2017 American Control Conference (ACC)*, pages 2680–2686, Seattle, WA, USA, May 2017. IEEE.

[6] Shadi Tasdighi Kalat, Siamak G. Faal, and Cagdas D. Onal. A Decentralized, Communication-Free Force Distribution Method With Application to Collective Object Manipulation. *Journal of Dynamic Systems, Measurement, and Control*, 140(9):091012, September 2018.

[7] Juekun Li, Wee Sun Lee, and David Hsu. Push-Net: Deep Planar Pushing for Objects with Unknown Physical Properties. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018.

[8] K.M. Lynch, H. Maekawa, and K. Tanie. Manipulation And Active Sensing By Pushing Using Tactile Feedback. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 416–421, Raleigh, NC, 1992. IEEE.

[9] Matthew T. Mason. Mechanics and Planning of Manipulator Pushing Operations. *The International Journal of Robotics Research*, 5(3):53–71, September 1986.

[10] Nikos Mavrakis, Amir M. Ghalamzan E., and Rustam Stolkin. Estimating An Object's Inertial Parameters By Robotic Pushing: A Data-Driven Approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9537–9544, Las Vegas, NV, USA, October 2020. IEEE.

[11] Nikos Mavrakis and Rustam Stolkin. Estimation and exploitation of objects ' inertial parameters in robotic grasping and manipulation: A survey. *Robotics and Autonomous Systems*, 124:103374, February 2020.

[12] Sean McGovern, Huitan Mao, and Jing Xiao. Learning to Estimate Centers of Mass of Arbitrary Objects. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1848–1853, November 2019. ISSN: 2153-0866.

[13] Sean McGovern and Jing Xiao. Learning and Predicting Center of Mass through Manipulation and Torque Sensing. In *2022 8th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, pages 60–66, Munich, Germany, February 2022. IEEE.

[14] Robert Nowak. Generalized binary search. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 568–574, Monticello, IL, USA, September 2008. IEEE.