Cooperative Transport in Mobile Robots

Aadesh Neupane[1]

Abstract— **Social animals cooperatively transport object which is many times bigger than themselves effectively. Mimicking those behaviors on real robots will have diverse applications in engineering, health care and search and rescue. In this paper, we define different categories of cooperative transport problems and discuss different tools and techniques to tackle them. We then show that occlusion-based cooperative transport techniques are effective when the object is convex and there are enough agents to overcome frictional force. Results show that even with only two robots, the occlusion-based technique is able to transport objects 60% of the time.**

I. INTRODUCTION

We see examples of simple solutions to complex problems in Nature. Simple organisms like bacteria without any central control or organizational hierarchy have evolved quite well to solve problems, survive and reproduce. Ants also have evolved collective behaviors for different tasks like foraging, nest defense, path planning and construction. These large group of insects like bees, ants and others are known as Swarms. Among different tasks these swarms can do, we focus on "Cooperative transport", a classical problem studied in robotics. "Cooperative transport" arises when a group of agent works together to carry a large object and form a consensus on a travel direction. Since a single agent doesn't have the resources to carry the large object, it needs to cooperative with its group to carry or move that object from a start location to a goal location. Performing "cooperative transport" with robots with the efficiency and robustness as of ants has wide applications in disaster relief, construction and search and rescue.

In nature, ants and termites have a robust solution to this problem of cooperative transport. Ants use a recruitment based strategy to gather a large number of ants to the location of interest. Then, without centralized control, they form a consensus on a common direction to move the object of interest. They do it without any knowledge of object shape or mass and are able to transport complex irregularly shaped objects. This allows the ants to collectively achieve tasks that individuals cannot achieve with added benefits of faster transport speeds and robustness to individual failure. Mimicking these behaviors would allow simple robots to transport heavy objects efficiently. Researchers have tried different strategies to mimic these behaviors onto robots. Robots have different kinematics based on their structure. So different type of robots will apply different techniques to solve the cooperative transport problem. Moreover, each solution or technique will be applicable to a particular type of robot with their own set of constraints.

In this paper, we explore different strategies employed to solve the cooperative transport problem and at the end describe an experiment with Cozmo robots based on occlusion-based cooperative transport [1].

II. OVERVIEW

The cooperative transport problem has been extensively studied in the context of multi-robot cooperation. Based on the literature, we can categorize the most common techniques as; 1) pulling, 2) pushing, and 3) caging.

a) Pulling: In nature, most of the cooperative transport is done by pulling. Pulling techniques involves connecting robots to the object using robotic manipulators for grasping or lifting. For pulling, robots need to know the shape and size of the object in prior to figure out the attach point on the object. Since, in many cases, objects attributes are not known a prior, pulling strategies are hard to carry out in practice.

Moreover, robotic manipulators add complexity to the existing base dynamics of the mobile robots which makes it hard to design a robust controller. Also, these robots are mostly non-holonomic i.e. the controllable degree of freedom is less than the total degree of freedom. For example, a robot car has 3 degrees of freedom i.e its position in x and y and its orientation. However, there are only two controllable degrees of freedom which are acceleration and turning angle of the steering wheel. Thus there are lots of constraints to design controllers for non-holonomic systems.

Thus the complexity of modeling physical mechanisms makes pulling strategy difficult to realize in practical robots.

b) Pushing: Pushing techniques are relatively simple as they only need to push the object of interest with a force greater than downward gravitation force and frictional force. When the collective force of the robots is greater than the opposing force on the object, the object starts to move. The direction and speed of the object depend on the resultant force acting on the large object. Since the pushing forces are distributed over multiple contact points on the object, the direction of movement is stable and increasing the number of the agents will increase the speed of the moving object.

In many cases, pushing might be a good technique for cooperative transport but there is the risk of wear and tear on the object being transported. Since the object is being pushed in excess of frictional force, there is the danger of object deformation. Thus this technique is not applicable for transporting fragile objects.

[1]Aadesh Neupane is graduate student, Brigham Young University aadeshnpn@byu.edu

c) Caging: Caging techniques are similar to the formation control problem. In formation control, robots organize themselves into a formation and maintains it during motion. For caging, robots organize themselves into a formation around the object of interest so that the object is trapped inside the formation. As long as the formation is maintained, the motion in robots will also induce motion in the trapped object. Caging can be applied to both pushing and pulling techniques.

Formulating the caging problem as formation control problem makes it easier in some cases but complexity arises when the object to be caged is of irregular shape. In addition, different caging techniques require a varying number of robots and additional information about object features. Thus, in practice, it is difficult to design caging techniques to accommodate both the numbers of robots and the objects features.

III. Modeling

For modeling the cooperative transport problem, we first need to build a model for our robots. If we consider homogeneous robots, the same model will work for all robots but when we consider M heterogeneous robots, we need upto M different models for the robots. Once we have our robot models, we need to choose among the three techniques described in section II. For this project, we will focus on pushing techniques as it is easy to analyze and implement on actual robots. Finally, we need a way to incorporate the communication between the robots. Mostly, researchers use graph theoretic methods to model the robots communication. Based on the communication behavior (implicit or explicit) the system analysis will differ.

In the section below, we look at two models based one based on implicit communication III-A and the other based on explicit communication III-B. The models for the agent will be introduced in the Section V-A.1 and V-B.1. Also, most of the multi-agent problems can be build on top of the agreement problem. The agreement problem in multi-agent systems relates to agreeing a global values for the state variables. So, in this section we will show examples related to the agreement problem and build up a model for cooperative transport in later section.

A. Explicit communication

For a general multi-agent system, Let N be the number of agents with its own dynamics and can communicate with other agents. Let equation 1 represent the dynamics of the agents where $i \in 1,2,3\cdots N$

$$\begin{aligned} \dot{x}_i &= Ax_i + Bu_i \\ y_i &= Cx_i + Du_i \end{aligned} \tag{1}$$

Now based on the communication structure, the dynamics of overall system evolve differently and can be represented by Linear Fraction Transformation (LFT).

Lets consider an example for agreement protocol. Lets these N agents be interconnected via relative information-exchange links. The rate of change of each unit's state is assumed to be governed by the sum of its relative states with respect to a subset of other neighboring agents. To visualize this lets consider four agents as show in Figure 1. Also the first order dynamics for each agent is outlined in equation 3.

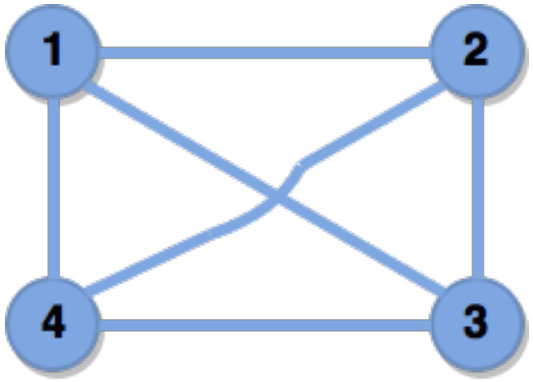

Fig. 1. Agreement protocol among four agents

$$\begin{aligned} \dot{x}_1 &= (x_2 - x_1) + (x_3 - x_1) + (x_4 - x_1) \\ \dot{x}_2 &= (x_1 - x_2) + (x_3 - x_2) + (x_4 - x_2) \\ \dot{x}_3 &= (x_2 - x_3) + (x_1 - x_3) + (x_4 - x_3) \\ \dot{x}_4 &= (x_2 - x_4) + (x_3 - x_4) + (x_1 - x_4). \end{aligned} \tag{2}$$

The equation 3 can be reduced to the general form

$$\dot{x}_i(t) = \sum_{j \in N(i)} (x_j(t) - x_i(t)), i = 1,2...N. \tag{3}$$

Now the above system can be represented as follows:

$$\dot{x}(t) = -L(G)x(t), \tag{4}$$

where $L(G)$ is the laplacian of the agents' interaction network G and $x(t) = (x_1(t),...,x_N(t))^T \in \mathbb{R}^N$

For a fully connected multi-agent system, the $L(G)$ is a positive semidefinite matrix. In cases where the connection is sparse or not fully connected, we can find the laplacian matrix as

$$L(G) = \Delta(G) - A(G), \tag{5}$$

where $\Delta(G)$ is the degree matrix and $A(G)$ is the adjacency matrix.

Based on the connectivity of the graph : directed vs undirected, balanced vs unbalanced, and strongly vs weakly connected, there are different ways to design the controller using graph decomposition and factorization tools.

For mobile robots, the graph is dynamic as the neighbored of an agent changes with time. Its harder to model a large number of robots using this approach. So, this approach will be useful for a few agents using the pulling strategy in cooperative transport. These agents need to be able to communicate both the mount location and their respective direction.

B. Implicit communication

Its more intuitive to employ implicit communication for the cooperative transport problem. When human or ants cooperate to carry a heavy load, they sense the resultant force and direction on the object and act accordingly. In this case, agents doesn't maintain direct communication with each other. Rather, they sense the activity of other agents through the medium at hand (the transport object).

IV. Centralized/ Decentralized

We can design both centralized and decentralized controllers to achieve cooperative transport. For centralized control, we have to gather the states from all the robots onto a single machine and make error adjustments and send appropriate input signals to all the robots. In the practical application for swarms and multi-agent system, using centralized control brings in many limitations. Most of the limitations induced in centralized control are due to communication issues like bandwidth, range, and latency. Also, accurate positioning and tracking systems are required to keep track of each states variables of the robots. Due to these limitations, it's better to have a decentralized controller. In case of decentralized control, each robot has its own controller to drive its states towards stability along with occasional control signals from other robots. In this paper, we focus on decentralized control as it is easier to implement on real robots.

V. Experiments

We compare and contrast two experiments on cooperative transport [1], [2] from the literature. Each uses implicit communication to transport an object to the goal location. Then we implement the methods described in the Chen et. al. [1] using Cozmo robots.

A. Simple Swarm

In this section, we describe the mathematical model and experiments from Michael et. al [2]. We then conclude with the limitations of this research and why we choose not to implement this model.

Michael et. al research is based on the specific version of cooperative transport known as distributed planar manipulation, where robots transport a lightweight object of known shape along a pre-specified global trajectory with control of both object orientation and total force. Since the global trajectory and shape is known, it is easy to analyze the solution. The problem with this approach is that it requires robots to track and estimate both the objects and the other robots. This limits its applicability and scalability. Instead of robots knowing global information, robots can sense locally and act locally which makes the controller very simple and overall system robust to noise.

Cooperative transport generally includes two steps: 1) agents randomly search the environment for the object to be transported and move towards the object to be in close proximity and 2) agents apply force to the object to transport it to the goal. Michael et. al consider to model only the second part of the step.

1) Model: The object of interest is modeled as an arbitrary connected 2D shape, with N agents applying force to it at arbitrary points. The object has mass M_o, and can slide with coefficients of static and kinetic friction μ_s, μ_k respectively. The system is assumed to be quasi-static which implies that the friction forces dominate inertial forces. Thus we treat the system as a first order system neglecting inertial terms. Assume that each agent a_i applies a force $\vec{f_i}$ to the object in any direction, independent of how or where it is attached. This assumption doesn't hold for most of the robots as they have non-holonomic movement. The research uses omni-directional grippers to hold the assumption of the force vector. Let V_{max} be the agent's maximum speed, f_0 be the magnitude of the maximum force the agent can apply and $C = f_0/V_{max}$ is a constant value for the agent. Then, the force applied by each agent $\vec{f_i}$ is dependent on the agent's velocity $\vec{v_i}$ as follows

$$\vec{f_i} = C(V_{max} - \vec{v_i} \cdot \hat{d})\hat{d}, \tag{6}$$

where $\hat{d}$ is a unit vector in the direction the agent is applying force. Intuitively, Equation 6 means that when the agent is moving with maximum velocity it can't apply any force i.e $\vec{v_i} = V_{max}$, implies $\vec{f_i} = 0$ and when the agent is stationary it can apply maximum force. Using this model, they show that when the resultant force from all the agents $\vec{F}_{tot}$ overcomes the $\vec{F}_{fric}$ frictional forces, the agents will drive the object to the goal location i.e the solution converges. Lets expand the above model to understand the translation and rotational effects on the object due to the resultant force.

a) Translation: From rigid body dynamics, the force acting on the surface is the same as the force acting on the center of the mass. The agent forces in the direction of the goal are opposed by friction $\vec{F}_{fric}$, so the total force on the object is given by

$$\vec{F}_{tot} = \sum_{i=1}^{N} \vec{f_i} - \vec{F}_{fric}. \tag{7}$$

If the object is not rotating then its velocity is the same at all points i.e $\vec{v_i} = \vec{v}_{obj}$ where $\vec{v}_{obj}$ is the center of mass velocity of the object. Then the total force from all agents can be written as

$$\sum_{i=1}^{N} \vec{f_i} = NC(V_{max} - \vec{v}_{obj}.\hat{d})\hat{d}. \tag{8}$$

Since the direction of $\vec{v}_{obj}$ and $\hat{d}$ is same, their dot product gives the norm which can be represented as $\|\vec{v}_{obj}\|$. Then we can write Equation 8 as

$$\sum_{i=1}^{N} \vec{f_i} = NC(V_{max} - \|\vec{v}_{obj}\|)\hat{d}. \tag{9}$$

The condition for the robots to overcome static friction and move an object initially at rest is

$$NC(V_{max} > \mu M_o g. \tag{10}$$

The steady-state velocity $\vec{v}_{ss}$ for a moving object is that at which the kinetic friction $\vec{f}_{fric} = -\mu_k M_o g\hat{d}$ balances the

forces exerted by the agents. Then,

$$\|\vec{v}_{obj}\ = \|\vec{v}_{ss}\| = \left(V_{max} - \frac{\mu_k M_o g}{NC}\right). \tag{11}$$

Thus the steady-state speed $\|\vec{v}_s s\|$ at which N agents can transport an object approaches the maximum speed of V_{max} asymptotically as N increases.

b) Rotation: The derivation of the translation component in the above section assumes that object translation is much faster than rotation. The negligible affect of rotation can be justified by showing that for any initial configurations, the object maximum rotation is 180^o in a particular direction. This result can be obtained using the quasi-static assumption. It can be shown that there exits a stable equilibrium within a rotation of no more that 180^o.

The main contribution of Michael et. al is to develop a simple swarm model with scalability properties. They show the model is stable and show that this strategy is agnostic to the object shape, location of object center-of-mass, attachment location of the agents, and number of agents using the Kilobot robot platform. The limitation of this work is the assumption that robots can apply force in any direction no matter the point of attachment. Thus, the object of interest needs to be fitted with omni-direction grippers. Attaching these special grippers on the object a prior is not always possible thus this technique is very limited to certain robots with homolomic moment.

B. Occlusion Based swarm

In this section, we describe the mathematical model and experiments from Chen et. al [1] and discuss our implementation details with Cozmo robot.

The main strategy behind this research is to push the object positioned so the agents line of sight to the goal is occluded by the object. There are two main phases for this strategy: 1) agents observing the goal and object location and move to the position behind the object where the goal location is occuled, and 2) push the object towards the normal vector of the center-of-mass of the object. The overall strategy is represented by a state machine with five states where each state has a different controller.

1) Model: This occlusion based strategy only works for object larger than the agents with arbitray convex objects.

Let $c \in \mathbb{R}^2$ be the center of mass of the a rigid convex object and let $g = [0,0]^T$ be the goal location. Let the perimeter of the object be described by a closed, convex, and differentiable curve given by

$$\mathbf{p}(\theta) = \begin{bmatrix} r(\theta)cos\theta \\ r(\theta)sin\theta \end{bmatrix} + \mathbf{c} \tag{12}$$

with $\theta \in [0, 2\pi]$ and $r : [0, 2\pi] \to \mathbb{R}$ is differentiable and satisfies $r(2\pi) = r(0)$. By specifying $r(\theta)$, any convex shape can be approximated by $\mathbf{p}$ and initially, $\mathbf{g}$ is outside of $\mathbf{p}$. The inward pointing normal vector on $\mathbf{p}(\theta)$, named $\mathbf{N}(\theta)$, is the derivative of $\mathbf{p}(\theta)$ rotated by $\frac{\pi}{2}$.

$$\mathbf{N}(\theta) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{p}'(\theta) \tag{13}$$

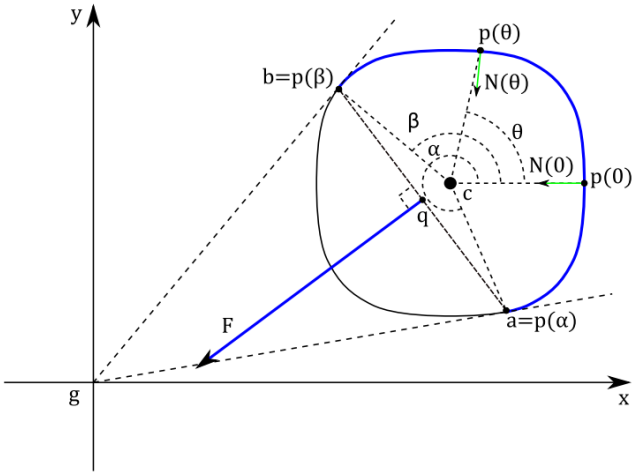

Fig. 2. Modeling the occlusion problem

Points along $\mathbf{p}$ where the direct line of sight to $\mathbf{g}$ are occluded between the two tangent points of $\mathbf{p}$ from $\mathbf{g}$. We write the two tangent points as $\mathbf{p}(\alpha)$ and $\mathbf{p}(\beta)$,for $\alpha, \beta \in [0, 2\pi]$. As tangent points, they satisfy

$$\begin{aligned} \mathbf{p}(\alpha).\mathbf{N}(\alpha) = \mathbf{p}(\beta).\mathbf{N}(\beta) = 0 \\ \mathbf{p}(\theta).\mathbf{N}(\theta) > 0, \forall \theta \in (\alpha, \beta). \end{aligned} \tag{14}$$

Since $\mathbf{p}$ is convex and $\mathbf{g}$ is outside $\mathbf{p}$, α and β are well defined. Let $a = \mathbf{p}(\alpha)$ and $b = \mathbf{p}(\beta)$, They satisfy

$$\begin{aligned} a_x c_y - a_y c_x > 0 \\ b_x c_y - b_y c_x < 0 \end{aligned} \tag{15}$$

,

where x and y denote the x and y coordinates. All of the terms defined so far are represented in the Figure 2. From the figure, all points $\mathbf{p}(\theta)$ with $\theta \in (\alpha, \beta)$ are on the occluded perimeter of the object, while all other points on p are visible from $\mathbf{g}$. Following are the lemmas described in the original work. For proofs see [1].

lemma 1: Assume that $n \to \infty$ robots are uniformly distributed along the occluded perimeter of the object and they are the only robots asserting a force on the object. The direction of the resultation force asserted on the object by the robots is equal to the direction of the vector $(\mathbf{b} - \mathbf{a})$ rotated by $\frac{\pi}{2}$ and its magnitude is proportional to $\|\mathbf{b} - \mathbf{a}\|$. i.e

$$\mathbf{F} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} (\mathbf{b} - \mathbf{a}) \tag{16}$$

lemma 2: If the combined force contributed by the robots $\mathbf{F}$ is considered as a single force, while Q is the torque induced by $\mathbf{F}$, the mid point of segement $\mathbf{ab}$ is an affecting point of $\mathbf{F}$.

As the object moves, $\mathbf{a}$ and $\mathbf{b}$ change over time. Assuming that the robots react to changes to fill the occluded space uniformly while pushing the object then Equation 16 is valid as long as $\mathbf{g}$ is outside $\mathbf{p}$. i.e

$$\mathbf{F(t)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} (\mathbf{b(t)} - \mathbf{a(t)}). \tag{17}$$

Then the translation dynamics of the center of mass of the object are

$$\mathbf{v} = \dot{\mathbf{c}}, \dot{\mathbf{v}} = \frac{\mathbf{F}}{M}, \tag{18}$$

where $\dot{\mathbf{v}}, \dot{\mathbf{c}}$ are the derivatives of $\mathbf{v}$ and $\mathbf{c}$ repsectively with time t, and M is the object's mass. Based on quasi-static assumption if the motion of rigid object is slow, then the translation dynamic becomes

$$\dot{\mathbf{c}} = k\mathbf{F}, \tag{19}$$

where $k \in \mathbb{R}^+$ is a positive constatnt that transfers $\mathbf{F}$ proportionally to the velocity of the object.

Now, we will show that based on the lemmas and dynamics of the system, the system converges i.e the object will arrive at the goal location as $t \to \infty$.

Theorem 1: When the dynamics is governed by Equation 19, then when $t \to \infty$, $\mathbf{g}$ will be on the object perimeter $\mathbf{p}$.

Proof: Let $l(t) = c(t).c(t)$ be the squared distance of the center of mass $\mathbf{c}$ to goal $\mathbf{g}$, then its derivative with regard to time is

$$\dot{l} = 2kc \cdot \mathbf{F}. \tag{20}$$

Then using the value of $\mathbf{F}$ from Equation 17, we get

$$\mathbf{c} \cdot \mathbf{F} = (b_x c_y - b_y c_x) - (a_x c_y - a_y c_x). \tag{21}$$

According to Equation 16, $\mathbf{c} \cdot \mathbf{F} < 0$. Hence, $\dot{l}(t)$ is strictly decreasing. Since $l(t) \geq 0 \forall t > 0$ (as long as $\mathbf{g}$ is outside $\mathbf{p}$), we get $\lim_{t\to\infty} = L \in \mathbb{R}$. Therefore,

$$\lim_{t\to\infty} \mathbf{c} \cdot \mathbf{F} = \lim_{t\to\infty} (b_x c_y - b_y c_x) - (a_x c_y - a_y c_x). \tag{22}$$

Comparing with Equation 15 this implies that,

$$\begin{aligned} \lim_{t\to\infty} (b_x c_y - b_y c_x) &= 0 \\ \lim_{t\to\infty} (a_y c_x - a_x c_y) &= 0. \end{aligned} \tag{23}$$

In other words, the areas of traingles **gca** and **gcb** approach zero as $t \to \infty$. Since $\mathbf{c}$ is always inside $\mathbf{p}$, the traingles **gca** and **gcb** can never have 0 area unless $a = g$ and $b = g$. Hence, as $t \to \infty$, $\mathbf{g}$ will be on $\mathbf{p}$. Thus, the object will ultimately coincide with the goal and stop moving. ■

C. Cozmo

1) Differential drive dynamic: We verified the stability of the occlusion based cooperative transport. We implemented the dynamics using a differential drive robot. We choose Cozmo robots 4 for implementation. We first describe the kinematics of the differential drive robots and then describe the state machine based controller that performs cooperative transport. The state machine based controller is decentralized. Figure 3 shows a simple model for a differential drive robot where v_l and v_r is the velocity of left and right wheel respectively, R is the radius of the wheel and L is the distance between two wheels. The states of the differential drive model are x and y which give the location of the robot and ϕ which is the orientation of the robot. In order to drive the

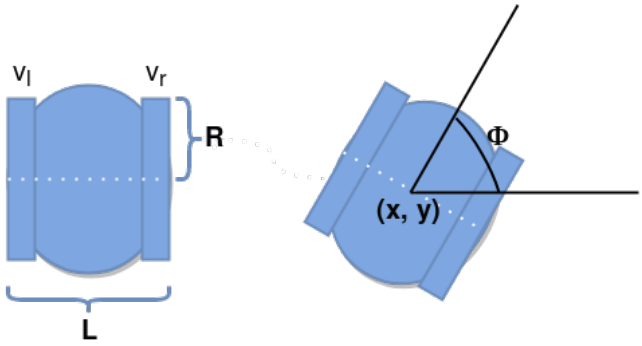

Fig. 3. Differential drive robot model

Fig. 4. Cozmo robot

robot, we need to connect v_l and v_r to the states of the robots which gives the kinematics equation for the robot [3].

$$\begin{aligned} \dot{x} &= \frac{R}{2}(v_r + v_l)\cos\phi, \\ \dot{y} &= \frac{R}{2}(v_r + v_l)\sin\phi, \\ \dot{\phi} &= \frac{R}{2}(v_r + v_l) \end{aligned} \tag{24}$$

Its cumbersome to map the rate of change of wheel velocity to the location of the robot. Also building controllers to adjust different wheel velocities to get desired location of the object is hard in practice. So, we will use the unicycle model for our robots. In this model, we will control translational v and angular velocity w to drive the robots. Equation 25 gives us the unicycle model.

$$\begin{aligned} \dot{x} &= v\cos\phi, \\ \dot{y} &= v\sin\phi, \\ \dot{\phi} &= w \end{aligned} \tag{25}$$

Comparing equation 24 and 25, we get

$$\begin{aligned} v = \frac{R}{2}(v_r + v_l) &\Rightarrow \frac{2v}{R} = v_r + v_l \\ w = \frac{R}{2}(v_r - v_l) &\Rightarrow \frac{wL}{R} = v_r - v_l \end{aligned} \tag{26}$$

and

$$v_r = \frac{2v + wL}{2R}$$
$$v_l = \frac{2v - wL}{2R} \tag{27}$$

While designing the controller 6 we will consider v and w to be inputs and these inputs will be mapped to v_r and v_l to drive the robot around.

2) State machine based controller: Cooperative transport involves different phases. For each phase a different controller is required. So, we utilize a state machine based controller which will switch the controller based on the different state value. Figure 5 show the different controllers and the conditions under which they operate.

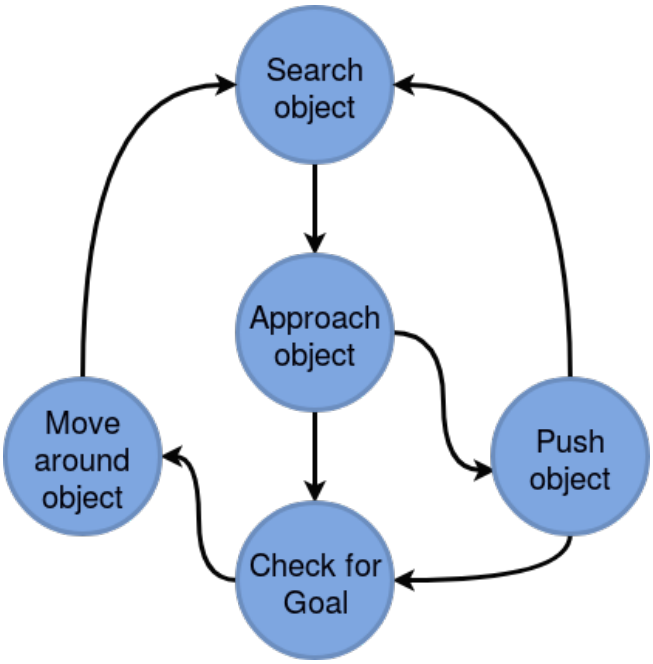

Fig. 5. Differential drive robot model

Our controller is the motion controller shown in Figure 6. To accomplish the goal of driving the robot to a desired linear velocity v_d and angular velocity w_d, our first step is to compute the error between the true velocities and the desired ones. So, let $e_v = v - v_d$ and $e_w = w - w_d$ be respectively the linear and angular velocity errors. Then a simple proportional control law is of the form

$$e_{am} = -K_{p1} e_V$$
$$e_{ad} = -K_{p2} e_w \tag{28}$$

drives the errors to approximately zero. For stronger convergence, a intergrator can be used along with the propotional term as

$$e_{am} = -K_{p1} e_v - K_{i1} \int_0^t e_v(\tau) d\tau$$
$$e_{ad} = -K_{p2} e_w - K_{i2} \int_0^t e_w(\tau) d\tau. \tag{29}$$

Suppose we want the robot to be in $p_r(t) = (x_r(t), y_r(t), \theta_r(t))$ a desired reference position, at time t. In order to drive the robot to the reference location, we need to have desired linear and angular velocities (v_d, w_d) to force the position of the robot $p = (x, y)$ to converge to the reference position $p_r = (x_r, y_r)$. The error between current position and reference position is

$$e = R(p_r - p),$$
$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{30}$$

When the e goes to zero then p converges to p_r. Then a PI controller is given by

$$\begin{bmatrix} v_d \\ w_d \end{bmatrix} = C \begin{bmatrix} v_r \\ w_r \end{bmatrix} - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{31}$$

where $C = diag\{cos(\theta_r - \theta), 1\}$, $u_1 = -k_1(x_r - x)$, $u_2 = k_2 v_r \sin(\theta_r - \theta)(y_r - y) - k_3(\theta_r - \theta)$ and each k_i is a positive gain constant. Equation 31 gives the control law to steer the robot to goal location.

a) Search Object: For this controller, we don't have any reference point to go to or track. So we implement a controller where the robot's motion is random. This can be achieved by sending random bounded values for v_l and v_r. During this random movement, if the robot see the object, then it will estimate the object's pose. Pose estimation is based on finding correspondences between points in the real environment and their 2D image projection. As pose estimation in wild is difficult problem to solve, it is common to aid it with synthetic markers. Among those synthetic markers, the use of binary square fiducial markers is quite popular. *Aruco* tags are a class of square fiducial markers composed by a wide black border and inner binary matrix which determines its identifiers. The black border facilitates its fast detection in the image and the binary codification allows its identification and the application of error detection and correction techniques.

A predefined size of those markers are pasted on the object. Now the robot can estimate the object pose with respect to itself. For each robot, the starting pose with always at origin i.e $(x, y, \Phi) = (0, 0, 0)$. Thus, for each robot the location of object and goal location varies based on their initialization. The robot randomly searches the environment until it registers both the goal and transport object in the environment. To realize the occlusion in practice, the robot must find a randevu point behind the object of interest. The randevu point is computed based on the equation of straight line.

b) Approach Object: After registering the location of goal and transport object, the robots need to travel to a randevu point. There could be obstacles while traveling, so we implemented a obstacle avoidance controller. Since there are no other sensors to know about surrounding, we implemented obstacle avoidance controller using optical flow using KLT tracker [4]. Then appropriate control signals v_l and v_r is generated to drive the robot to the randevu point. After that, the robots aligns it orientation to the object. A

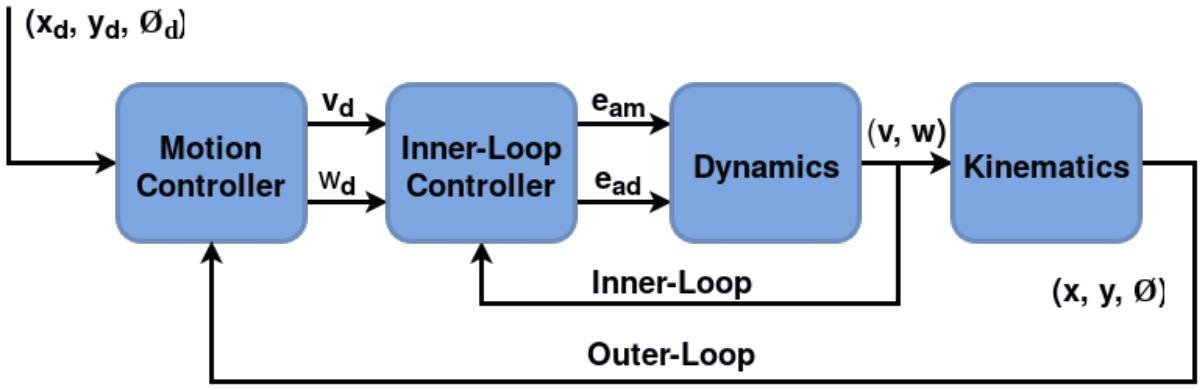

Fig. 6. Control loop for the robots

simple PID controller then drives the robot towards the object of interest.

c) Push Object: When the robot is in the close proximity of the object it starts to move towards the direction of goal. Since the object is heavy, there should be enough agents to push the object to overcome frictional force. Until, the goal is reached, the robots push the object. If N is large then there is higher probability of robots collectively moving the object to the goal location.

d) Check for Goal: Based on the pose estimation of goal location from *Search object* step, the robots check if they have reached the goal location. If not, they continue pushing the object.

e) Move around object: During motion, if the robot slips from the occluded region, it will readjust its position to maintain itself in the occluded region.

3) Results: For the experiments, we used only 2 cozmo robots [5]. Since only two robots were present, only 60% successful cooperative transport were observed. Even though the location obtained through pose estimation were noisy, the controller was able to achieve cooperative transport with only 2 robots. Thus, the controller has properties of robustness. The codes[1] and the videos[2] of the experiments has been published online.

VI. CONCLUSIONS

This paper discussed the stability criteria of occlusion based cooperative transport and occlusion based technique was implemented on real robots. Results demonstrated that only with 2 robots the technique performed well achieving hitrate of 60%.

ACKNOWLEDGMENT

I would like to thank Dr. Warnick for approving this project. Also, I would like to thank Charles Johnson for early review of this paper.

[1] https://github.com/aadeshnpn/cooperative-transport

[2] https://www.youtube.com/watch?v=WvW2b_O350E

REFERENCES

[1] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, "Occlusion-based cooperative transport with a swarm of miniature mobile robots," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 307–321, 2015.

[2] 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*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 47–54.

[3] R. Carona, A. P. Aguiar, and J. Gaspar, "Control of unicycle type robots tracking, path following and point stabilization," 2008.

[4] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.

[5] C. Anki. Cozmo robot. [Online]. Available: https://github.com/anki/cozmo-python-sdk