Herding by Caging: a Topological Approach towards Guiding Moving Agents via Mobile Robots

Anastasiia Varava*, Kaiyu Hang†, Danica Kragic*, and Florian T. Pokorny*
* Robotics, Perception, and Learning Lab, Centre for Autonomous Systems, School of Computer Science and Communication, Royal Institute of Technology (KTH), Stockholm, Sweden
{varava, dani, fpokorny}@kth.se
† Robotics Institute and Institute for Advanced Study, Hong Kong University of Science and Technology, Hong Kong
kaiyuh@ust.hk

***Abstract*—In this paper, we propose a solution to the problem of *herding by caging*: given a set of mobile robots (called herders) and a group of moving agents (called sheep), we move the latter to some predefined location in such a way that they cannot escape from the robots while moving. We model the interaction between the herders and the sheep by assuming that the former exert virtual "repulsive forces" pushing the sheep away from them. These forces induce a potential field, in which the sheep move in a way that does not increase their potential. This enables the robots to partially control the motion of the sheep. We formalize this behavior geometrically by applying the notion of *caging*, widely used in robotic grasping. We show that our approach is provably correct in the sense that the sheep cannot escape from the robots. We propose an RRT-based motion planning algorithm, demonstrate its probabilistic completeness, and evaluate it in simulations.**

I. Introduction and Related Work

In this work, we propose an approach towards guiding mobile agents called *herding by caging*. We are interested in the problem of driving the motion of a group of mobile agents, which we call *sheep*, by a team of robots. Robots in the driving team that provides driving forces are called *herding robots*. The driving forces imposed by the herding robots are modelled by a distance-based potential field, and we assume that the motion of the sheep is restricted by the forces imposed by the herding robots.

Motion planning for a team of mobile robots is an essential problem in many real world applications, such as the coverage control for mobile sensing networks [5], behavior-based control for robot teams [1], and communication-constrained motion planning for multi-robot systems [16], etc. In research on robot formation control and motion planning, the problem formulations can be classified into three groups [2, 10]: 1) A robot in a team is designated as the leader, and is first commanded to follow a predefined pose trajectory (position and orientation) to lead the team. The other robots in the team are moving by following the leader while having to satisfy a set of task-related geometric constraints [21]; 2) With the concept of virtual structure, the robot team is modeled as a single structure, in which the motion of each robot is translated from the desired global structure [7]; and 3) The robot team is desired to provide a group behavior, and each single robot's motion is subject to a weighted average of several behaviors [1].

However, to the best of our knowledge the research has been mainly focused on the control aspect of the formation maintenance and the motion planning for achieving the desired formations. The problem of how a team of mobile robots can interact with moving agents, such as a group of people or animals, is not widely addressed. In particular, we consider a scenario where a group of "active" agents (mobile robots) controls the motion of "passive" agents by exerting repulsive forces. The "passive" agents tend to keep away from the "active" ones, which is usually formalized by introducing a distance-based potential field induced by the virtual repulsive forces exerted by the latter.

The possible applications of our problem are not limited to herding animals, [13, 20, 23]. For instance, one can use several mobile robots working cooperatively to evacuate people during emergency situations, [9]. Moreover, one could use the robot team to either secure a team of people or to isolate a group of dangerous mobile objects, e.g., drones, from humans. A similar approach can also be applied to a team of robots to collect oil leaked on water, [3], or to keep animals away from the runways in airports etc., [13].

In [18, 23], the shepherding behavior of robots has been proposed for a small robot team (≤ 3 robots). The authors addressed the problem using genetic algorithms and neural networks to model the local behaviors of flock control. In [13], the authors provided the first work on shepherding behaviors with large flock size. In [20], the authors propose a self-propelled particle model of local attraction-repulsion type to model herding of a group of agents by one shepherd. The problem of herding cows using smart collars equipped with GPS and sound amplifiers was also considered in [4]. In [9] the authors work on guiding people in the environment represented as a potential field, which enables the authors to guide people in urban areas. Artificial potential field-based controllers for herding have also been used in [22, 11]. This approach is also analyzed in [19].

In our setting, the herding team forces the sheep to move from an initial state to a specified goal region in arbitrary maps in both 2D and 3D, while ensuring that the sheep do

not escape from the herders. We formulate the problem as a motion planning problem for the herding team. We adopt the concept of *caging* to formally describe the situation in which the motion of the sheep is restricted by the herders.

Caging is a way of restricting the mobility of an object without immobilizing it completely, [17]. For this, representations of physical obstacles and energy fields (such as gravity) can be utilized, [14]. So far, the notion of caging has been mostly used in robotic grasping. In [8], a caging-based approach to multirobot manipulation has been proposed. In [3], the authors use an approach to separate and manipulate sets of objects using cables.

We address the caging verification problem by computing homology groups of superlevel sets of the potential function to check that the sheep are located in a bounded region of low potential, and therefore cannot escape from the robots. We represent the superlevel sets as alpha complexes, [6]. Alpha complexes have been used before for proving caging (path non-existence), for instance, in [14, 15].

The motion process consists of two alternating phases. In the *repulsion* phase, the robots remain static, and the sheep move away from the herders. A potential function strictly monotonically decreases as the distance to the closest robot increases. The motion of the sheep is restricted by this potential field, as they aim to never increase their potential during the repulsion phase. Therefore, we can control their motion by surrounding them with the robots, so that the robots form a closed region of high potential around the flock. Since the sheep never increase their potential, they cannot escape from the herders as long as they are surrounded by a region of high potential.

However, in the *herding* phase, when both the sheep and the robots move, the sheep can move in any direction with a bounded velocity. We do not make any assumptions about the direction of their motion. As such, our system is partially controlled: we affect the motion of the sheep only in the repulsion phase.

The contributions in this work can be summarized as follows: first, we formulate the problem of guiding a group of mobile agents as *herding by caging*, and propose a rigorous caging verification method using techniques from computational topology. Second, we provide an RRT-based robot team motion planning algorithm using the above mentioned verification procedure, and demonstrate that the proposed algorithm is probabilistically complete. Finally, we implement our algorithm in a 2D-workspace and analyze the possibility of generalizing our approach to the 3D case.

II. Problem Formulation

A. Potential-based Caging

To address our problem, we first define the notion of a cage. According to the classical definition, an object is *caged* by a caging tool if it cannot escape arbitrarily far from the the caging tool (i.e., a manipulator in robotic grasping, or a group of robots in our context). In our setting, a sheep is caged if it is surrounded by a region of potential higher than its own, so that it would have to temporarily increase its potential in order to escape from the robots.

In this work, we call the passive agents "sheep", even though the possible applications are not limited to herding animals. The active agents are referred to as "herders". We abstract both the sheep and the robots as points. The goal of this paper is to define and propose a solution to the *cage-steering* problem: starting from initial position, where the flock is surrounded by the herders, we move the latter in such a way that the repulsive forces induced by them push the flock to some predefined location.

Consider a workspace $\mathcal{W} \subset \mathbb{R}^d$. Let $\mathcal{S} = \{s_1, s_2, .., s_m\}$ denote the set sheep, and $\mathcal{R} = \{r_1, r_2, .., r_n\}$ denote the set of robots controlling their motion. [1] We assume that the robots move in $\mathcal{W}$, and therefore the configuration space $\mathcal{C}$ of the team is a subset of $\mathcal{W}^n$. Let us denote the collision-free subset of $\mathcal{C}$ by $\mathcal{C}_{free}$.

Let $\mathrm{d}_c : \mathcal{W} \to \mathbb{R}$ denote the distance from a point to closest robot from the set $\mathcal{R}$. Assume that the sheep tend to keep away from the robots. To formally describe their behaviour, we introduce a potential function $p_c : \mathcal{W} \to \mathbb{R}$.

Definition 1: A potential function $p_c : \mathcal{W} \to \mathbb{R}$ is a continuous function strictly monotonically decreasing with the distance from $x \in \mathcal{W}$ to the closest robot from $\mathcal{R}$, when the robots are at the configuration $c \in \mathcal{C}_{free}$.

Since the potential of any point $x \in \mathcal{W}$ is uniquely defined by the distance to the closest robot, it is convenient to use the following notation: $p_d(\mathrm{d}_c(x)) = p_c(x)$, where $p_d : \mathbb{R}_{\geq 0} \to \mathbb{R}$ and $p_c : \mathcal{W} \to \mathbb{R}$

B. The Motion Model

Let us now model the interaction between the sheep and the robots. We consider two types of behavior: *repulsion* and *herding*. These two types represent two different phases of the steering process. During the repulsion phase, the robots do not move, while the sheep can move with respect to them. During the herding phase, both the robots and the sheep move. In practice, the motion process consists of a sequence of alternating repulsion and herding phases. We assume that the velocity of the sheep never exceeds v_s. Let us first describe the repulsion phase.

1) Repulsion Phase: Let $s(t)$ denote the trajectory of a sheep in $\mathcal{W}$ during some period of time $t \in [0, T]$. Assume that the robots stay at the same configuration $c \in \mathcal{C}_{free}$. Then the sheep never moves to points of higher potential:

$$\forall t_1 < t_2 \in [0, T] : \; p_c(s(t_1)) \geq p_c(s(t_2)).$$

Note that this assumption allows a more general class of motion than just following the gradient of the potential field, although the latter is a valid example. This also implies that the position of the robots does not uniquely define the motion of the flock, but rather partially restricts it.

[1] The number of robots n does not depend on m, but cannot be smaller than 3 in the case of a 2D workspace.

2) Herding Phase: During herding phase, the robots move between two configurations $c_0, c_1 \in \mathcal{C}_{free}$. Even though the sheep might want to keep away from the herders during this phase as well, this might be infeasible due to several reasons. First, they may not be able to accurately predict the motion of the robots, and therefore might accidentally move closer to them. Second, the computation of an optimal trajectory requires a good sense of orientation and fast reaction (in the applications where by "sheep" we mean people or animals), and a certain computational capacity (when the passive agents are also robots). Moreover, their perception of the herding robots' positions might not be precise due to noise. To keep our setting as generic as possible, we do not make any assumptions on the direction of the sheep during this phase, and only assume that the velocity of the sheep is bounded.

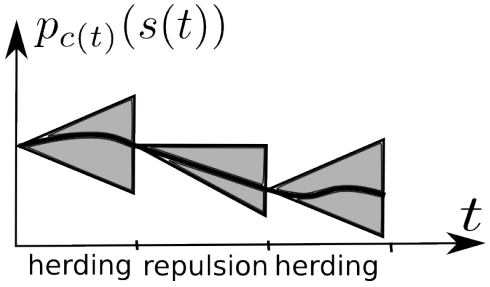

Fig. 1. Three alternating motion phases are depicted in this figure. The black curve reflects the actual dynamics of the potential value of a single sheep s – $p_{c(t)}(s)$, while the grey triangles correspond to the regions of possible values. The angle of the grey cones is determined by the sheep's maximal velocity v_s.

Note that while the potential value of a sheep does not increase during the repulsive phase, it might increase during the herding phase, see Fig. 1. This happens because during the herding phase the robots do not control the motion of the sheep, which makes the system only *partially controlled*. For this reason, the herding phase is limited in time in such a way that we can guarantee that during it the sheep will not escape from the robots, which is possible as the sheep move at a constant velocity. In contrast, when the robots do not move (i.e., during the repulsion phase) the sheep move in a way that never increases their potential. Therefore, if the sheep are properly surrounded by a region of high potential, they never escape from the robots, and hence we do not have to limit the duration of this phase.

III. Mathematical Background

In this section, we explain the concepts from computational topology we use later in the paper.

A. Holes, Voids and Homology Groups

Algebraic topology aims to classify topological spaces up to continuous deformations. One of its main tools is the computation of homology groups. Intuitively, the dimension of the i^{th} homology group $H_i(\mathcal{X})$ of the space $\mathcal{X}$ represents the number of $i-$dimensional "voids" in $\mathcal{X}$. For instance, in a 2D space the elements of $H_1(\mathcal{X})$ correspond to the "holes", while in a 3D space $H_2(\mathcal{X})$ represents its "voids".

In this paper, we are interested in computing homology groups in two cases. When we consider a two-dimensional workspace $\mathcal{W}$, we construct its subset $\mathcal{X}$ in such a way that the potential value at any of its points is higher then in its complement $\mathcal{W}-\mathcal{X}$. This way, the sheep are caged when they are located in bounded subsets of lower potential, which can be seen as "holes" in $\mathcal{X}$. We compute the first homology group of $\mathcal{X}$ with coefficients in $\mathbb{Z}_2$ to find these "holes". The elements of $H_1(\mathcal{X})$ are closed curves going "around" the "holes", i.e., the curves that cannot be continuously deformed into a point in $\mathcal{X}$. Each "hole" corresponds to a homology class. Since these curves have high potential values, they bound the mobility of the sheep located in the "holes", see Fig.2.

In the case of a three-dimensional workspace $\mathcal{W}$, the bounded subsets of low potential correspond to the "voids" in the high potential subset $\mathcal{X}$ of $\mathcal{W}$, which are represented by its second homology group $H_2(\mathcal{X})$. Similarly to the two-dimensional case, the elements of $H_2(\mathcal{X})$ are subsets of $\mathcal{X}$, separating the "voids" from the remaining part of $\mathcal{W}-\mathcal{X}$.

B. Simplicial Complexes

For computational reasons, it is convenient to work with discrete versions of spaces, which can be achieved by representing them as *simplicial complexes*.

A geometric $k-$simplex $\sigma = [v_0, .., v_k]$ in $\mathbb{R}^d$ is a convex hull of $k+1$ ordered affinely independent elements $v_0, .., v_k \in \mathbb{R}^d$. A convex hull of a subset of $\{v_0, .., v_k\}$ is called a face τ of the simplex σ, which is denoted by $\tau \leq \sigma$. A finite simplicial complex $\mathcal{K}$ is a non-empty set of simplices such that:

- if $\tau \leq \sigma$, then $\tau \in \mathcal{K}$,
- if $\sigma, \sigma' \in \mathcal{K}$, then $\sigma \cap \sigma' = \emptyset$ or $\sigma \cap \sigma' \in \mathcal{K}$

In 2D, a simplicial complex $\mathcal{K}$ is a set of vertices, line segments and triangles, whose intersections are either empty or belong to $\mathcal{K}$. In 3D, a simplicial complex consists of vertices, segments, triangles and tetrahedra with the same property.

C. Topology of a Union of Balls: Alpha Complexes

In the next section, we deal with subsets of $\mathcal{W}$ represented as unions of closed balls of a fixed radius. In particular, we will be interested in the homology groups of these sets. Let $X = \{x_1, .., x_n\}$ be a finite set of points in $\mathbb{R}^d$, and let $R > 0$ be a real number. Consider a union of closed balls with centers at points from X and radii R.

From the computational point of view, it is not easy to compute homology groups of $\bigcup_{i=1}^n B_R(x_i)$ directly. Fortunately, this is not necessary, as we can work with its discrete version – the *alpha complex*.

An alpha complex $A(R)$ corresponding to the union of balls $\bigcup_{i=1}^n B_R(x_i)$ is a simplicial complex with vertices $\{x_1, .., x_n\}$ which lies strictly inside $\bigcup_{i=1}^n B_R(x_i)$, and is homotopy equivalent to the latter [6].

Given a set of points $X = \{x_1, .., x_n\}$, we can continuously increase the radius and get a nested family of unions of balls. Correspondingly, we get a nested family of alpha complexes, $\emptyset = A(R_0) \subset A(R_1) \subset .. \subset D(X)$, where $D(X)$ is

the Delaunay triangulation of X. Any alpha complex is a subcomplex of Delaunay triangulation of X, and since the latter is finite, the family of nested subcomplexes is also finite. In our work, we use this fact for cage verification.

IV. Methodology

We want to guarantee that the sheep are caged during repulsion phases, and that the herding phases are limited in time in such a way that the sheep do not escape from the robots. In this section we provide the necessary definition and formalize potential-based caging. Then, we derive sufficient conditions for safe moves between two caging configuration – i.e., conditions for the robots' motion during the herding phase. Later, we provide the algorithm for motion planning. From now on, we assume that $\mathcal{W}$ is two-dimensional. We discuss the possible generalization to the three-dimensional case later in this section.

Definition 2: Let $\mathcal{R}_c$ denote the set of robots in configuration c. A *high potential fence* $HPF(\mathcal{R}_c)$ induced by robots $\mathcal{R}$ in configuration c is a closed curve in $\mathcal{W}$, such that there exists a non-empty open set O in its interior [2], the supremum of the potential value $\sup_{x\in O}(p_c(x))$ in which is strictly lower than the potential of the fence, defined as $p_c(HPF(\mathcal{R}_c)) = \min_{z\in HPF(\mathcal{R}_c)} p_c(z)$.

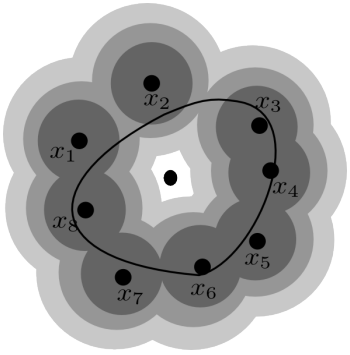

Fig. 2. The black curve is a high potential fence induced by the robots $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$. Different shades of grey depict different superlevels of the potential function. The choice of the set O is not unique: e.g., any open set from the white and light grey regions from the interior of the curve satisfies the required conditions.

Fig. 2 illustrates the concept introduced above. Note that in some situations we do not need to use all the robots from $\mathcal{R}$ to form a high potential fence. Note also that some configurations can induce several high potential fences.

Definition 3: A configuration $c \in \mathcal{C}$ is a *caging configuration* if there exists a high potential fence induced by $\mathcal{R}_c$.

Definition 4: A sheep s *is caged* by the robots if it is situated in the interior of some high potential fence, and its potential is strictly lower than the potential of the fence.

A. Cage verification

Suppose we have an initial static configuration of robots and a set of sheep, and we want to check whether they form a cage. For that, we need to check whether the caging condition holds, i.e., if the robots form a high potential fence such that the sheep are located in its interior.

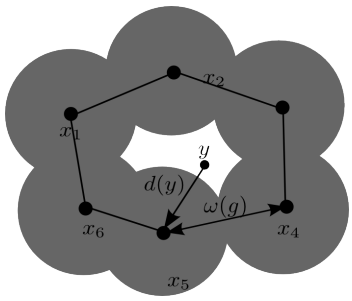

Fig. 3. This figure depicts a caging chain formed by 5 robots. $\omega(g)$ is the length of the longest segment of the chain. Grey circles depict the superlevel set of the potential function, containing the caging chain.

From the computational point of view, it is convenient to have a discrete version of the notion of high potential fence. Namely, we would like to deal with points and segments instead of continuous curves. We therefore introduce the following definition.

Definition 5: An ordered set of robots $(x_{a_1}, .., x_{a_k})$ that form a high potential fence around a sheep with coordinates y, together with segments connecting each pair of consecutive robots, and the first and the last ones, forms a polygonal chain g called *caging chain*.

A caging chain (see Fig. 3) is a special case of a high potential fence. The advantage of this notion is that a caging chain is a closed curve consisting of a finite number of segments, connecting pairs of robots. In other words, a caging chain g formed by k robots with coordinates $(x_{a_1}, x_{a_2}, x_{a_3}, .., x_{a_k})$ can be viewed as a set of pairs $g = \{(x_{a_1}, x_{a_2}), (x_{a_2}, x_{a_3}), .., (x_{a_k}, x_{a_1})\}$. When the robots move, the caging chain also moves and deforms; as long as the movement preserves the cage, we can keep track of the deformations of the initial caging chain. Assume that at some moment of time t the new coordinates of the corresponding robots are $(x^t_{a_1}, x^t_{a_2}, x^t_{a_3}, .., x^t_{a_k})$, and the cage has been preserved along the way. By g_t we denote the deformation of the corresponding caging chain, which now can be considered as $g_t = \{(x^t_{a_1}, x^t_{a_2}), (x^t_{a_2}, x^t_{a_3}), .., (x^t_{a_k}, x^t_{a_1})\}$.

The width of the caging chain is defined as the maximal distance from any point of the chain g to the set of robots forming it, which is the same as a half of the length of its longest segment: $\omega(g) = 1/2 \max_{(x_i,x_j)\in g}(\text{length}(x_i, x_j))$.

This value is important, since it defines the lowest potential of the caging chain: $p_c(g) = p_d(\omega(g))$.

Consider a sheep $s \in \mathcal{S}$ with coordinates y, and assume that the robots in configuration $c \in \mathcal{C}$ have coordinates $(x_1, x_2, .., x_n)$, respectively. To verify the caging condition, we first need to compute the high potential fences induced by c. For that, we study the topology of the superlevel sets of the potential function $p_c(y)$.

A superlevel set is a set of the form $L^+_q = \{x \in \mathcal{W} | p_c(x) \geq q\}$, where the value of the potential function is not lower than q. To check if a given configuration forms a cage, one can

[2]By interior of a closed curve in $\mathbb{R}^2$ we mean the union of bounded connected components in its complement.

consider the topological properties of the sets defined above. Namely, if a sheep with coordinates $y \in \mathcal{W}$ and a potential value $p_c(y) < q$ for some positive real number q lies in the interior of some closed curve $\phi \subset L_q^+$, then ϕ is a high potential fence by definition. Therefore, the sheep is caged.

Let first us assume that given the position of the sheep y and a real number $q > p_c(y)$, we want to check whether the sheep is caged by some high potential fence, whose potential value is not lower than q. For this it is enough to check whether there exists a closed curve $\phi \subset L_q^+$, whose interior contains y. First of all, observe that any such curve ϕ cannot be contracted to a point in L_q^+. Therefore, if $\phi \subset L_q^+$ contains y in its interior, then it represents a non-trivial class of the homology group of the space L_q^+. Moreover, if some other closed curve $\phi' \subset L_q^+$ is homotopy equivalent to ϕ (and therefore, corresponds to the same element in the first homology group), then ϕ' is also a high potential fence caging the sheep at the point y. Thus, it is enough to consider one closed curve per first homology class in L_q^+ to check if L_q^+ has a closed curve whose interior contains y.

Let us now interpret the shape of the set L_q^+ geometrically. Since by definition any potential function strictly monotonically decreases with the distance to the set of robots, for any $q > 0$ the set L_q^+ is in fact a union of closed balls of radius R centred at points $\{x_1, x_2, .., x_n\}$, where $p_d(R) = q$: $L_q^+ = \bigcup_{i=1}^n B_R(x_i)$.

This observation is crucial, as it enables us to consider discrete approximations of L_q^+ without losing any important information about the topological properties of the latter. Namely, consider an alpha complex $A(R)$ (see Section III for the introduction to alpha complexes). It is well-known ([6]) that $A(R)$ is homotopy equivalent to and lies strictly inside of $\bigcup_{i=1}^n B_R(x_i)$, and therefore preserves its topological properties of interest. Moreover, each first homology class representative of $A(R)$ whose interior contains y is a caging chain, as it consists of the vertices corresponding to the positions of robots, and links between them. Thus, the first homology group of $\bigcup_{i=1}^n B_R(x_i)$, can be extracted directly from $A(R)$.

For each first homology class representative[3] g of $A(R)$ we check if y lies inside its interior. If this is the case, then the sheep is caged by means of the caging chain g.

Let us now find *an optimal* high potential fence – i.e., a fence of the highest possible potential. Recall from Section II that we define the potential of the fence as the potential of its weakest point, $p_c(\phi) = min_{x \in \phi}(p_c(x))$. Therefore, we need to find the *maximum* value $q_{max} > p_c(y)$ for which there exists $\phi \subset L_{q_{max}}^+$ containing y in its interior. For that, we consider a family of sets $\mathcal{F} = \{L_{q_0}^+, L_{q_1}^+, .., L_{q_k}^+\}$, where $\infty = q_0 > q_1 > .. > q_k > p_c(y)$ and hence $L_{q_0}^+ \subset L_{q_1}^+ \subset .. \subset L_{q_k}^+$. Here the family $\mathcal{F}$ is finite, as explained in Sec. III.

Then q_{max} is the greatest value among $q_0 > q_1 > .. > q_k$ such that $L_{q_{max}}^+$ contains a closed curve whose interior contains y.

[3]We compute homology with coefficients in $\mathbb{Z}_2$

Let us now make an important observation:

Proposition 1: The caging chains computed as first homology class representatives of the corresponding alpha complex are optimal.

Proof: Let the sheep be located at point y, and the robots be at a configuration c. Let the sheep be caged, and let q_{max} be the potential value of the optimal high potential fence caging the sheep. Then, there is a closed curve $\phi \subset L_{q_{max}}^+$, such that $y \in \mathrm{int}(\phi)$. Recall that $L_{q_{max}}^+ = \bigcup_{i=1}^n B_R(x_i)$, where $p_d(R) = q_{max}$. Consider an alpha complex $A(R)$. Since $A(R)$ is homotopy equivalent to $\bigcup_{i=1}^n B_R(x_i)$, there exists a closed curve $\phi' \simeq \phi$, $\phi' \subset A(R) \subset \bigcup_{i=1}^n B_R(x_i)$. Therefore, ϕ', and any other closed curve from $A(R)$, homotopy equivalent to it, is an optimal high potential fence caging the sheep. ■

So, to check if the sheep with coordinates y is caged, we perform the following procedure, see Alg. 1. We construct a Delaunay triangulation $D(\{x_1, .., x_n\})$ based on the coordinates of the robots. Then, we build a sequence of superlevel sets (represented as alpha complexes) $\mathcal{F} = \{L_{q_0}^+, L_{q_1}^+, .., L_{q_k}^+\}$, where $q_k > p_c(y)$. If there is such a superlevel set $L_{q_{cage}}^+$, in which one of the first homology class representatives contains y in its interior, then the sheep is caged. If there are several such superlevel sets, we select the one with the largest potential value. In practise, this is the same as to construct a nested sequence of unions of closed balls with growing radii, starting with $R = 0$ and finishing once $R \geq \mathrm{d}(y)$. In its turn, the union of balls can be replaced with the corresponding alpha complex.

Algorithm 1: Cage verification

```
input : robots coordinates (x_1, .., x_n), sheep coordinate y
output: Δp
Δp = 0
D ← Delaunay((x_1, .., x_n))
foreach L_q^+ ⊂ D such that q > p_c(y) do
    L_q^+ ← AlphaComplex((x_1, .., x_n),d_c(y))
    G_1 ← FirstHomologyRepresentatives(L_q^+)
    foreach g ∈ G_1 do
        if IsInsidePolygon(y, g) then
            Δp ← q
            Break
        end
    end
end
return Δp
```

B. Safely Connected Cages

Recall that we make no assumptions about the trajectories of the sheep during the herding phase. Therefore, we need to derive some sufficient conditions to guarantee that the sheep will not escape from the robots as they move between two caging configurations c_1 and c_2. If there exists such a safe path between these two configurations, we call c_1 and c_2 *safely connected*. The following proposition tells us how the robots can move during the herding phase without letting the sheep escape far from them, see Fig. 4:

Proposition 2: Consider a potential function $p_c : \mathcal{W} \to \mathbb{R}$. Assume that the robots move from a caging configuration c_0 to another configuration c_T, and let $c(t)$ denote their trajectory

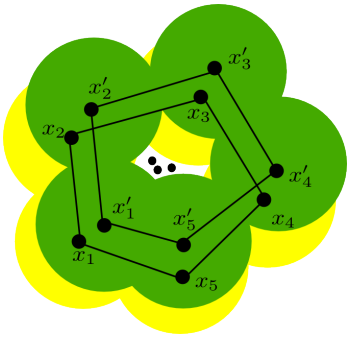

Fig. 4. This figure illustrates a typical motion of the robots during herding phase. The robots moves from $(x_1, x_2, x_3, x_4, x_5)$ to $(x'_1, x'_2, x'_3, x'_4, x'_5)$. Yellow circles depict the initial superlevel set, while the final one is depicted in green.

in $\mathcal{C}_{free}$, $t \in [0, T]$. Let s_t denote the trajectory of a sheep in $\mathcal{W}$, and assume that when $t = 0$ the sheep is caged by robots $\{r_1, .., r_k\} \subseteq \mathcal{R}$. Finally, let g_0 be a caging chain formed by $\{r_1, .., r_k\}$, $s_0 \in \operatorname{int} g_0$, and let g_t denote its deformation at time t, induced by the corresponding movement of the robots.

Then the sheep never escapes the cage for any $t \in [0, T]$, provided that

1) both the sheep and the robots team move with constant non-zero velocities, v_s and v_r;
2) $\mathrm{d}_{c(0)}(s_0) - \omega(g_0) > (v_s + 2 \cdot v_r) \cdot T$

Proof: We need to demonstrate that for any moment of time $t \in [0, T]$

- the potential of the sheep is lower than the potential of the caging chain, $p_{c(t)}(s_t) < p_{c(t)}(g_t)$, and
- the sheep is inside the caging chain, $s_t \in \operatorname{int} g_t$.

Note that the latter statement follows from the former, provided $s_0 \in \operatorname{int} g_0$. Therefore, it is enough to show that for any $t \in [0, T]$

$$p_{c(t)}(s_t) < p_{c(t)}(g_t). \tag{1}$$

From the first condition of the Proposition, for any $t \in [0, T]$ we have

$$\mathrm{d}_{c(t)}(s_t) \geq \mathrm{d}_{c(0)}(s_0) - t \cdot v_s - t \cdot v_r.$$

Therefore, since p strictly monotonically decreases with the distance to the robots, we have

$$p_{c(t)}(s_t) \leq p_d(\mathrm{d}_{c(0)}(s_0) - t \cdot v_s - t \cdot v_r).$$

Similarly, for $t \in [0, T]$ we have

$$p_{c(t)}(g_t) \geq p_d(\omega(g_0) + t \cdot v_r)$$

To prove 1, it is enough to show that

$$p_d(\omega(g_0) + t \cdot v_r) > p_d(\mathrm{d}_{c(0)}(s_0) - t \cdot v_s - t \cdot v_r), \tag{2}$$

which by strict monotonicity of $p()$ follows from

$$\mathrm{d}_{c(0)}(s_0) - t \cdot v_s - t \cdot v_r > \omega(g_0) + t \cdot v_r, \tag{3}$$

which can be written as

$$\mathrm{d}_{c(0)}(s_0) - \omega(g_0) > t \cdot v_s + 2 \cdot t \cdot v_r.$$

The latter is true for any $t < T$, since

$$\mathrm{d}_{c(0)}(s_0) - \omega(g_0) > (v_s + 2 \cdot v_r) \cdot T$$

■

Remark 1: Instead of expressing these conditions in terms of time and velocity, we can as well assume that the time T is fixed. This gives us a condition on the acceptable displacement of the robots, and we then can adjust the velocity v_r to satisfy the necessary conditions from the above proposition. Let $\varepsilon_r = v_r \cdot T$ and $\varepsilon_s = v_s \cdot T$. In our implementation, we make sure that $\mathrm{d}_{c(0)}(s_0) - \omega(g_0) > \varepsilon_s + 2 \cdot \varepsilon_r$. Assuming that the velocity of robots is higher than the velocity of sheep, it is enough to check that $\mathrm{d}_{c(0)}(s_0) - \omega(g_0) > 3 \cdot \varepsilon_r$.

C. Motion planning

Consider now the steering problem. Assume that we have an initial caging configuration c_{init}. We need to move the flock to the specified goal region without breaking the cage. This means that each configuration in the path must form a cage, and all the subsequent pairs of cages must be safely connected. Let U be a set of all possible actions which we can apply to a given configuration in order to move to the next one. Namely, each $u \in U$ specifies a collection $\{v_1, .., v_n\}$ of vectors defining a movement of the system. When u is applied, the i^{th} robot moves from x_i to $x_i + v_i$. For simplicity, we discretize U by assuming that each robot can be moved along some vector $e_{k\Delta\phi}$ at a distance $||m\Delta d||$, where $\Delta\phi$ and Δd are the discretization steps. By $\mathcal{C}_{cage} \subset \mathcal{C}_{free}$ we denote the subspace containing all caging configurations.

1) Standard RRT – a naive approach: We start with a simple approach to motion planning – standard RRT with rejection sampling. We sample random configurations from $\mathcal{C}$, and check whether they are *(i)* collision-free, and *(ii)* form valid cages of the required size. If both conditions hold, we then would search for the nearest vertex in the RRT, and make a motion from it towards the randomly selected configuration, so that the nearest and the new vertices would be *(i)* reachable from each other by a straight collision-free motion, and *(ii)* safely connected. However, this approach does not perform well in practice, as the probability of randomly sampling a caging configuration in $\mathbb{R}^{2n}$ is small. Therefore, instead of sampling random configurations from $\mathcal{C}$, we aim to sample new configurations biased towards the caging subset of the configuration space, $\mathcal{C}_{cage}$, see Alg.2.

2) RRT-based caging planner: Namely, we generate samples based on those caging configurations which we already have. We randomly choose an existing configuration c from the RRT, and with probability 0.5 we apply one of the following operations to generate new configuration c_{rand}: *Random translation:* in this case, c_{rand} has the same shape as c (i.e., relative positions of the robots are preserved), but is moved from c in a random direction to random distance. *Random perturbation:* in this case, we randomly choose a direction $\phi_i \in (0, 2\pi)$ for each robot, $i \in \{1, .., n\}$, and move the i^{th} robot in the chosen direction at a distance $\varepsilon(c)$, where $\varepsilon(c)$ depends on the location of the closest sheep and the width

of the caging chain, and is defined in such a way that c_{rand} is also a caging configuration.

Algorithm 2: RRT-based caging planner

input : configuration space $\mathcal{C}$, robots initial configuration c_{init}, sheep coordinates f, workspace $\mathcal{W}$
output: RRT $\mathcal{T}$

$\mathcal{T} \leftarrow \emptyset$
if *CageVerification* (c_{init}, f) **then**
  $\mathcal{T} \leftarrow \{c_{init}, f\}$
  $t_{start} \leftarrow$ CurrentTime()
  while $time < MaxTime$ **do**
    c, $f \leftarrow$ RandomFromTree($\mathcal{T}$)
    $\delta \leftarrow$ Rand(0, 1)
    if $\delta < 0.5$ **then**
      $c_{rand} \leftarrow$ RandomTranslation(c)
    else
      $c_{rand} \leftarrow$ RandomPerturbation(c)
    end
    if *Extend*($\mathcal{T}$, c_{rand}, $\mathcal{W}$) = *Reached* **then**
      break
    end
    $time \leftarrow$ CurrentTime() - t_{start}
  end
end
return $\mathcal{T}$

Once we have a new random configuration c_{rand}, we are trying to extend the RRT, see Alg. 3. For that, we find the nearest existing configuration c_{near} in the tree, and then with probability $1 - \rho$ we apply the action from U that leads us as close as possible to c_{rand}. Alternatively, with probability $\rho > 0$ we apply to c_{near} a random action instead of the optimal one. This is done for the sake of preserving probabilistic completeness in our modification of RRT. However, in practice our algorithm is more efficient when ρ is close to 0.

Algorithm 3: Extend

input : RRT $\mathcal{T}$, configuration c_{rand}, workspace $\mathcal{W}$
output: $Reached$ or $Extended$ or $Failed$

$c_{near}, f \leftarrow$ NearestNeighbor($\mathcal{T}$, c_{rand})
$\delta \leftarrow$ Rand(0, 1)
if $\delta < \rho$ **then**
  $c_{new}, f' \leftarrow$ ApplyRandomAction($\mathcal{T}$, c_{near}, f, c_{rand})
else
  $c_{new}, f' \leftarrow$ ApplyTheBestAction($\mathcal{T}$, c_{near}, f, c_{rand})
end
if *AddVertex*($\mathcal{T}$, c_{new}, f', c_{near}, f, $\mathcal{W}$) **then**
  if $c_{new} \in \mathcal{C}_{goal}$ **then**
    return $Reached$
  else
    return $Extended$
  end
else
  return $Failed$
end

3) Motion verification during the herding phase: As a result of applying an action to c_{near}, we get a new configuration c_{new} which we would like to add to the tree. At this point we need to make sure that c_{near} and c_{new} are safely connected, and there is a straight-line collision-free path between them, see Alg. 4. If there is no straight-line collision-free path, we compute the closest reachable configuration on the way to c_{new} instead. If it is a caging configuration and lies at a sufficient distance from c_{near}, and c_{near} and c_{new} are safely connected (by Prop. 2), we add it to the tree.

Algorithm 4: AddVertex

input : RRT $\mathcal{T}$, configuration to be added c_{new}, sheep coordinates corresponding to the new configuration f', nearest configuration in the tree c_{near}, sheep coordinates within nearest configuration f, workspace $\mathcal{W}$
output: $True$ or $False$

$\mathcal{M} \leftarrow$ GetLinearMotion(c_{near}, f, c_{new}, f', $\mathcal{W}$)
c_{new}, $f' \leftarrow$ LastBeforeCollision(c_{near}, f, $\mathcal{M}$, $\mathcal{W}$)
return *CageVerification*(c_{new}, f') **and** (*Distance*(c_{near}, c_{new}) $> \varepsilon$) **and** *AreSafelyConnected*(c_{near}, c_{new}, f)

The correctness of our approach is enforced by two verification procedures: *CageVerification*(c_{new}, f') and *AreSafelyConnected*(c_{near}, c_{new}, f). The former corresponds to the caging verification during the repulsion phase and is described in details in Alg. 1. The latter checks the safety during the herding phase, see Alg. 5. Here, the function *GetMaxRobotMovement*(c_0, c_1) computes ε_r from Remark 1 – the length of the paths between the initial and the resulting positions of the robots. *DistanceToConf*(s_0, c_0) returns $d_{c_0}(s_0)$, and *ConfChainWidth*(c_0) returns $\omega(g_0)$.

Algorithm 5: AreSafelyConnected

input : Two configurations c_0 and c_1, sheep coordinates s_0 corresponding to the first configuration
output: $True$ or $False$

$\Delta \leftarrow$ GetMaxRobotMovement(c_0, c_1)
return *DistanceToConf*(s_0, c_0)$-$*ConfChainWidth*(c_0) $> 3\Delta$

4) Probabilistic completeness: Let us now discuss probabilistic completeness of the proposed algorithm. Assume that there is a sequence of configurations $c_1, .., c_q$, such that $c_1 = c_{init}$, and $c_q \in \mathcal{C}_{goal}$. Assume also that there exists a sequence of actions $u_1, .., u_{q-1}$ that when applied to c_1 yields the sequence $c_2, .., c_q$. Here all of the configurations are in the same connected component of $\mathcal{C}_{cage}$, and each pair of subsequent configurations are safely connected.

Then we can formulate the following proposition (the proof is almost identical to the proof of Theorem 3 from the work by LaValle and Kuffner, [12], and we briefly recall it here).

Proposition 3: The probability that the above proposed algorithm initialized with c_{init} will contain a configuration from $\mathcal{C}_{goal}$ tends to one as the number of iterations goes to infinity.

Proof: Assume the RRT contains c_i as a vertex after some finite number of iterations. Consider the Voronoi diagram associated with the RRT vertices. Since in our implementation no two RRT vertices lie within a specified $\varepsilon > 0$ of each other, $\mu(\mathrm{Vor}(c_i)) > 0$. At each iteration of the algorithm, there is a non-zero probability p_1 that c_i will be selected as the nearest vertex. Since U is finite, and we select a random action from U with positive probability ρ, there is a non-zero probability p_2 that the right action u_i leading us to the new configuration c_{i+1} will eventually be selected. We apply this argument iteratively from c_1 to c_{q-1}. ■

D. Generalization: 3D workspace

Our algorithm can be naturally generalized to higher dimensions. Let now $\mathcal{W} \subset \mathbb{R}^3$. As before, a potential function

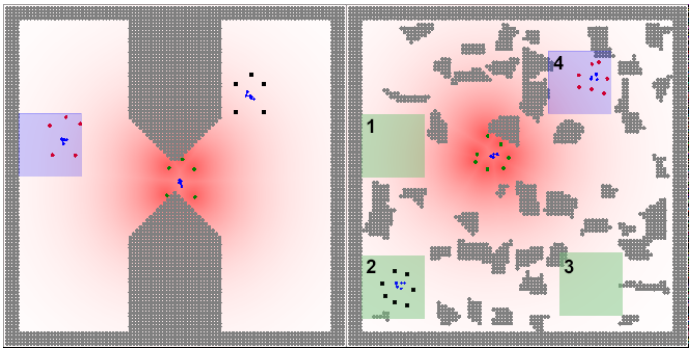

Fig. 5. The obstacles are depicted in grey; the goal region is the blue square; the black dots correspond to the initial configuration of the robots, the green dots depict the current configuration, and the red dots are the robots in the final configuration. The potential field induced by the current configuration is depicted in red. In the second environment, the 3 remaining goals are depicted in green.

$p : \mathcal{W} \to \mathbb{R}$ has to be continuous and strictly monotonically decreasing as the distance from the point to the set of robots increases, and the same assumption regarding the motion of the flock is made. A high potential fence can be considered as a high potential subset HPS of $\mathcal{W}$, such that at least one connected component of $\mathcal{W} - HPS$ is bounded. Proposition 1 can be directly generalized, as the dimensionality of the space does not play a crucial role in the proof. Moreover, the same motion planning algorithm can be applied with only slight technical modifications.

V. Experiments

In this section, we consider two types of workspaces and run our algorithm different number of robots. We run our experiments on an Intel i7 CPU laptop with 12 GB of RAM.

A. Narrow passages

In our first experiment, we consider a family of simple rectangular workspaces containing two polygonal obstacles and a narrow passage between them (Fig. 5, left side). Our goal is to see how the efficiency of our algorithm depends on $d(\mathcal{R})/w(\mathcal{C})$, where $w(\mathcal{C})$ denotes the width of the narrow passage, and $d(\mathcal{R})$ is the maximal distance between any two robots in the initial configuration. In other words, we would like to see how well the robots can change the shape of their formation without breaking the cage, when they have to go through narrow passages. We perform experiments with 4, 5, 6, and 7 robots respectively. We consider 20 random initial configurations and run the algorithm 50 times for each of them. We interrupt it after 30 seconds in case the solution has not been found. We compute the average execution time in seconds, as well as the success rate (the percentage of successful runs). Since this workspace is very simple, and the main difficulty is to find the path through the narrow passage, we consider only one goal region, and all the initial configurations are located on the opposite side of the workspace.

The above table presents the average computation time of the successful runs, as well as the success rate. We see that for $d(\mathcal{R})/w(\mathcal{C})$ between 0.8 and 1.0 our algorithm performs quite well. However, we can observe that as we decrease the width of the passage, the success rate significantly decreases. This likely is the case because our algorithm cannot explore the caging space well enough within 30 seconds, as the shapes of the new configurations are biased towards those which are already in the tree.

$\frac{d(\mathcal{R})}{w(\mathcal{C})}$	4 robots	5 robots	6 robots	7 robots
1.0	1.7 s, 96%	2.0 s, 91%	3.1 s, 89%	4.3 s, 79%
0.9	2.8 s, 89%	2.9 s, 87%	1.5 s, 77%	4.3 s, 76%
0.8	2.9 s, 79%	3.3 s, 86%	1.8 s, 57%	2.7 s, 71%
0.7	3.7 s, 73%	3.5 s, 62%	3.3 s, 44%	9.9 s, 44%
0.6	7.1 s, 45%	2.9 s, 26%	3.5 s, 20%	8.6 s, 26%

B. Random polygons

In the second experiment, we consider a more complex environment (Fig. 5, right side). As before, we analyze how well our algorithms perform depending on the number of robots: 4, 5, 6, and 7. We consider 4 different goal regions and 20 random initial configurations. We run the algorithm 50 times for each combination of the start and the goal. We interrupt it after 30 seconds in case the solution has not been found.

Goal	4 robots	5 robots	6 robots	7 robots
1	6.9 s, 87%	7.5 s, 91%	7.3 s, 80%	6.5 s, 86%
2	7.0 s, 87%	5.2 s, 90%	10.5 s, 69%	9.9 s, 96%
3	17.1 s, 16%	5.2 s, 1%	6.3 s, 2%	5.9 s, 2%
4	2.9 s, 96%	5.3 s, 90%	7.4 s, 89%	4.7 s, 91%

The above table presents the average computation time of the successful runs, as well as the success rate. This experiments shows the performance of our algorithm in a cluttered environment. Here, we can see that the Goal 3 is practically unreachable within 30 seconds. This happens because it is densely surrounded with obstacles, and therefore the shape of the configurations in the path should be significantly different from the shape of the initial one.

VI. Conclusion

In this paper, we proposed an approach towards *herding by caging*: given a set of mobile robots in the two-dimensional workspace, we partially control the set of moving objects (called sheep) by means of the potential field induced by repulsive forces exerted by the robots.

We address the caging verification problem using a classical tool from algebraic topology – homology groups. We propose an RRT-based algorithm for path planning and show that it preserves an important advantage of RRT – probabilistic completeness. We implement our algorithm and analyze its performance in experiments.

In the future work we would like to run the experiments in the three-dimensional settings. We also plan to run some real world experiments with mobile robots moving in both two-dimensional and three-dimensional settings.

VII. Acknowledgment

This work has been supported by the Knut and Alice Wallenberg Foundation and Swedish Research Council. The authors thank O. Kravchenko for his help with the implementation, and J. A. Haustein and J. F. Carvalho for their valuable feedback.

References

[1] T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[2] R. W. Beard, J. Lawton, and F. Y. Hadaegh. A coordination architecture for spacecraft formation control. *IEEE Transactions on Control Systems Technology*, 9(6):777–790, 2001.

[3] S. Bhattacharya, S. Kim, H. Heidarsson, G. S. Sukhatme, and V. Kumar. A topological approach to using cables to separate and manipulate sets of objects. *The International Journal of Robotics Research*, 34(6):799–815, 2015.

[4] Z. Butler, P. Corke, R. Peterson, and D. Rus. Virtual fences for controlling cows. In *International Conference on Robotics and Automation*, volume 5, pages 4429–4436. IEEE, 2004.

[5] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[6] H. Edelsbrunner and J. Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[7] M. Egerstedt and X. Hu. Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951, 2001.

[8] J. Fink, M. A. Hsieh, and V. Kumar. Multi-robot manipulation via caging in environments with obstacles. In *International Conference on Robotics and Automation*, pages 1471–1476. IEEE, 2008.

[9] A. Garrell, A. Sanfeliu, and F. Moreno-Noguer. Discrete time motion model for guiding people in urban areas using multiple robots. In *International Conference on Intelligent Robots and Systems*, pages 486–491. IEEE, 2009.

[10] S. Garrido, L. Moreno, and P. U. Lima. Robot formation motion planning using fast marching. *Robotics and Autonomous Systems*, 59(9):675–683, 2011.

[11] V. Gazi and K. M. Passino. A class of attractions/repulsion functions for stable swarm aggregations. *International Journal of Control*, 77(18):1567–1579, 2004.

[12] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[13] J. M. Lien, S. Rodriguez, J.-P. Malric, and N. M. Amato. Shepherding behaviors with multiple shepherds. In *International Conference on Robotics and Automation*, pages 3402–3407, 2005.

[14] J. Mahler, F. T. Pokorny, S. Niyaz, and K. Goldberg. Synthesis of energy-bounded planar caging grasps using persistent homology. *Workshop on the Algorithmic Foundations of Robotics*, 2016.

[15] Z. McCarthy, T. Bretl, and S. Hutchinson. Proving path non-existence using sampling and alpha shapes. In *International Conference on Robotics and Automation*, pages 2563–2569. IEEE, 2012.

[16] G. A. S. Pereira, A. K. Das, R. V. Kumar, and M. F. M. Campos. Decentralized motion planning for multiple robots subject to sensing and communication constraints. *Departmental Papers (MEAM). University of Pennsylvania*, 2003.

[17] A. Rodriguez, M. T. Mason, and S. Ferry. From caging to grasping. *The International Journal of Robotics Research*, 31(7):886–900, 2012.

[18] A. Schultz, J. J. Grefenstette, and W. Adams. Roboshepherd: Learning complex robotic behaviors. In *In Robotics and Manufacturing: Recent Trends in Research and Applications, Volume 6*, pages 763–768. ASME Press, 1996.

[19] M. Schwager, D. Rus, and J.-J. Slotine. Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. *The International Journal of Robotics Research*, 30(3):371–383, 2011.

[20] D. Strömbom, R. P. Mann, A. M. Wilson, S. Hailes, A. J. Morton, D. Sumpter, and A. J. King. Solving the shepherding problem: heuristics for herding autonomous, interacting agents. *Journal of The Royal Society Interface*, 11(100):20140719, 2014.

[21] H. G. Tanner. Iss properties of nonholonomic vehicles. *Systems & Control Letters*, 53(3-4):229–235, 2004.

[22] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switching networks. *IEEE Transactions on Automatic control*, 52(5):863–868, 2007.

[23] R. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron. Experiments in automatic flock control. *Robotics and Autonomous Systems*, 31(1-2):109–117, 2000.