Multi-Robot Control Using Time-Varying Density Functions

Sung G. Lee, Yancy Diaz-Mercado,
and Magnus Egerstedt, *Fellow, IEEE*

***Abstract*—An approach is presented for influencing teams of robots by means of time-varying density functions, representing rough references for where the robots should be located. A continuous-time coverage algorithm is proposed and distributed approximations are given whereby the robots only need to access information from adjacent robots. Robotic experiments show that the proposed algorithms work in practice as well as in theory.**

***Index Terms*—Multi-robot teams, coverage control, time-varying density functions**

I. Introduction

Coverage control for multi-robot systems has received significant attention lately, and it is concerned with how to position agents in such a way that "surveillance" of a domain of interest is maximized. This is typically achieved by associating a density function to the domain, as was done in [1]–[6]. However, the focus of previous coverage algorithms has largely been on static density functions. This does not provide enough flexibility when human operators are to adaptively interact with a team through a dynamic re-shaping of the density functions, which is the topic under consideration in this paper.

To enable this line of inquiry, we require an algorithm that can guarantee multi-robot optimal coverage given general time-varying density functions. Applications to this beyond the means for multi-robot influence can be found in a number of domains. For example, in search and rescue scenarios, the density function could represent the probability of a lost person being at a certain point in an area, e.g., [7]. Additionally, optimal coverage of density functions for multi-robot surveillance and exploration was used in [5], where the density function was modeled to be a function of the explored "frontier." (For other examples, see [8] and references therein.) To date, relatively little work has been done on coverage with time-varying density functions. In [1], the time-varying case was investigated under a set of simplifying assumptions on the density functions, while in [4], the density functions were used as a means to tracking moving targets. While simulations and experiments verified that coverage was indeed achieved, formal guarantees were absent.

In contrast to [1] and [4], in this paper we derive an algorithm that guarantees optimal coverage, under certain assumptions which will be further clarified along the text, and in general outperformed previous proposed approaches in simulation and robotic implementation. The outline is as follows: In Section II the problem setup is discussed in the context of locational costs that evaluate how effective given robot configurations are at achieving coverage. This is followed by the formulation of the main, centralized algorithm for coverage with time-varying density functions in Section III. A decentralized approximation based on truncated Neumann series is given in Section IV, and the different algorithms are implemented and compared on five mobile robots in Section V.

The authors are with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA (e-mail: slee656@gatech.edu; yancy.diaz@gatech.edu; magnus@gatech.edu)

This work was supported by grant number FA9550-13-1-0029 from the US Air Force Office for Scientific Research.

II. Locational Costs and Voronoi Tessellations

In order to discuss about optimal coverage, one first has to associate a cost to a robot configuration that describes how well a given area is being covered. To do this we will follow the construction of this so-called locational cost, as was done, for example, in [1] and we stress that no results in this section are new – we simply include them for the sake of easy reference.

Let $D \subset \mathbb{R}^2$ be the two-dimensional convex domain representing the area of interest. Moreover, let $\phi : D \times [0,\infty) \to (0,\infty)$ be the associated density function, which we will assume is bounded and continuously differentiable in both arguments, and where $\phi(q,t)$ captures the relative importance of a point $q \in D$ at time t.

The coverage problem involves placing n robots in D, and we let $p_i \in D$, $i = 1,\dots,n$ be the position of the i^{th} robot. Moreover, the domain itself will be divided into regions of dominance, e.g., [2], $P_1,\dots,P_n$ (forming a proper partition of D), where the idea is to let robot i be in charge of covering region P_i. One can then ask how good the choice of p and P is, where $p = [p_1^T,\dots,p_n^T]^T$, and $P = \{P_1,\dots,P_n\}$. The final piece needed to answer this question is a measure of how well a given point $q \in D$ is covered by robot i at position $p_i \in D$ (see [9] and references therein). As the performance of a large class of sensors deteriorate with a rate proportional to the square of the distance, the resulting locational cost is

$$H(p,P,t) = \sum_{i=1}^{n} \int_{P_i} \|q - p_i\|^2 \phi(q,t) dq. \tag{II.1}$$

At a given time t, when a configuration of robots (p) together with the partition (P) minimize (II.1), the domain is said to be optimally covered with respect to ϕ. However, it is possible to view the minimization problem as a function of p alone, [1], by observing that given p, the choices of P_i that minimize (II.1) is

$$V_i(p) = \left\{ q \in D \;\middle|\; \|q - p_i\| \leq \left\|q - p_j\right\|, i \neq j \right\}.$$

This partition of D is a Voronoi tessellation – hence the use of V_i to denote the region. With this choice of region, we can remove the partition as a decision variable and instead focus on the locational cost

$$H(p,t) = \sum_{i=1}^{n} \int_{V_i(p)} \|q - p_i\|^2 \phi(q,t) dq \tag{II.2}$$

In [9], [10] it was shown that

$$\frac{\partial H}{\partial p_i} = \int_{V_i} -2(q - p_i)^T \phi(q,t) dq, \tag{II.3}$$

and since $\phi > 0$, one can define the mass m_i and center of mass c_i of the i^{th} Voronoi cell, V_i, as

$$m_i(p,t) = \int_{V_i(p)} \phi(q,t)dq,\ c_i(p,t) = \frac{\int_{V_i(p)} q\phi(q,t)dq}{m_i}. \quad \text{(II.4)}$$

Using these quantities, the partial derivative in Equation II.3 can be rewritten as

$$\frac{\partial H}{\partial p_i} = 2m_i(p_i - c_i)^T. \quad \text{(II.5)}$$

From this expression, we can see that a critical point of (II.2) is

$$p_i(t) = c_i(p,t), \quad i = 1,\ldots,n, \quad \text{(II.6)}$$

and a minimizer to (II.2) is necessarily of this form, [11]. Moreover, when Equation II.6 is satisfied, p is a so-called centroidal Voronoi tessellation (CVT).

The robots being in a CVT configuration does not, however, imply that the global minimum of (II.2) is attained, e.g., [9]. In fact, the CVT is in general not unique given a density function ϕ. Finding the globally minimizing configuration is a difficult problem due to the nonlinearity and nonconvexity of (II.2), as discussed in [12]. As such, in this paper, we are interested in designing algorithms that guarantee convergence to local minima with respect to time-varying density functions, and we make no claims about finding the global minimum.

In light of Equation II.5, the gradient direction (with respect to p_i) is given by $p_i - c_i$. As such, a (scaled) gradient descent motion for the individual robots to execute would be

Lloyd:

$$\dot{p}_i = -\kappa(p_i - c_i) \quad \text{(II.7)}$$

where κ is a positive gain. This is a continuous-time version of Lloyd's algorithm [13] for obtaining CVTs as long as ϕ does not depend on t. The way to see this, as was done in [2], is to take $H(p)$ in Equation (II.2) (note that we assume that H only depends on p and not on t for the purpose of this argument) as the Lyapunov function,

$$\frac{d}{dt}H(p) = \sum_{i=1}^{n} \frac{\partial}{\partial p_i}H(p)\dot{p}_i = -2\kappa\sum_{i=1}^{n} m_i \|p_i - c_i\|^2.$$

By LaSalle's invariance principle, the multi-robot system asymptotically converges to a configuration $\{\|p_i - c_i\|^2 = 0,\ i = 1,\ldots,n\}$, i.e., to a CVT, [2].

However, if ϕ is time-varying, the same control law does not stabilize the multi-robot system to a CVT. This point can be hinted at by investigating the evolution of a time-dependent $H(p,t)$,

$$\begin{aligned}\frac{d}{dt}H(p,t) &= \sum_{i=1}^{n} \frac{\partial}{\partial p_i}H(p,t)\dot{p}_i + \frac{\partial}{\partial t}H(p,t)\\ &= \sum_{i=1}^{n}\int_{V_i} \|q - p_i\|^2 \frac{\partial \phi}{\partial t}(q,t)dq - 2\kappa\sum_{i=1}^{n} m_i \|p_i - c_i\|^2.\end{aligned}$$

There is no reason, in general, to assume that this expression is negative since we do not want to impose assumptions on slowly varying, or even quasi-static, density functions. Instead, what is needed is a new set of algorithms for handling the time-varying case, which is the topic of the next section.

III. TIME-VARYING DENSITY FUNCTIONS

To get around the problem associated with non-slowly varying density functions, timing information must be included in the motion of the robots. In [1], this was done through the assumption that $\phi(q,t)$ is such that

$$\frac{d}{dt}\left(\sum_{i=1}^{n}\int_{V_i} \|q - c_i\|^2 \phi(q,t)dq\right) = 0.$$

Letting

$$\dot{m}_i = \int_{V_i} \dot{\phi}(q,t)dq, \qquad \dot{c}_i = \frac{1}{m_i}\left(\int_{V_i} q\dot{\phi}(q,t)dq - m_i c_i\right),$$

the algorithm in [1] for time-varying density functions is given by

Cortes:

$$\dot{p}_i = \dot{c}_i - (k + \frac{\dot{m}_i}{m_i})(p_i - c_i). \quad \text{(III.1)}$$

Under the previously mentioned assumption on ϕ, $H(p,t)$ again becomes a Lyapunov function when the agents move according to Equation III.1, and convergence to a time-varying CVT is established.

Unfortunately, the assumption required to make Equation III.1 work is rather restrictive and for the remainder of the paper, we will develop new methods for handling time-varying density functions that do not impose major assumptions on $\phi(q,t)$. In fact, if the density function is to be thought of as an external, human-generated input to the system, there are no a priori reasons why the human operator would restrict the interactions to satisfy particular regularity assumptions on ϕ.

One way forward is to note that if we are already at a CVT at time t_0, i.e., $p(t_0) = c(p(t_0),t_0))$, where $c = [c_1^T,\ldots,c_n^T]^T$, it should be possible to maintain the CVT. In other words, if we can enforce that

$$\frac{d}{dt}(p(t) - c(p(t),t)) = 0 \quad \forall t \geq t_0,$$

the time-varying CVT would have been maintained. This means that

$$\dot{p} = \dot{c} = \frac{\partial c}{\partial p}\dot{p} + \frac{\partial c}{\partial t},$$

which rearranges to

$$\dot{p} = \left(I - \frac{\partial c}{\partial p}\right)^{-1}\frac{\partial c}{\partial t}. \quad \text{(III.2)}$$

As such, we have established the following result

Theorem III.1. *Let $p(t_0) = c(p(t_0),t_0)$. If*

$$\dot{p} = \left(I - \frac{\partial c}{\partial p}\right)^{-1}\frac{\partial c}{\partial t},\ t \geq t_0$$

then

$$\|p(t) - c(p(t),t)\| = 0,\ t \geq t_0$$

as long as the inverse $(I - \partial c/\partial p)^{-1}$ is well-defined.

There are a number of issues that must be resolved about the evolution in Equation III.2, namely (i) When is the inverse well-defined?; (ii) How can one ensure that $p(t_0) =$

$c(p(t_0),t_0)$?; (*iii*) How is $\partial c/\partial p$ computed?; and (*iv*) Is it possible to implement this in a distributed manner? The first question is in general quite hard to answer. In [14] it was shown that in the time-invariant case, the inverse is well-defined as long as $\phi(p)$ is a log-concave function of p. Moreover, we need ϕ to be continuously differentiable in both arguments, and these two conditions are enough to ensure that the inverse exists. However, this is not particularly satisfying and it does indeed pose a major challenge to the ambition of providing algorithms that can handle general, time-varying density functions. As will be seen in Section IV, it is possible to get around this restriction while, at the same time, answer the fourth question through the introduction of a well-posed Neumann approximation of the inverse as a mechanism for achieving distributed versions of the algorithm. The answer to the remaining two questions will be discussed below.

The first issue to be addressed is the constraint that $p(t_0) = c(p(t_0),t_0)$ for some initial time t_0. This is, practically speaking, easily achievable by adding a proportional term that forces the robots to a CVT,

TVD-C:

$$\dot{p} = \left(I - \frac{\partial c}{\partial p}\right)^{-1}\left(-\kappa(p-c) + \frac{\partial c}{\partial t}\right). \tag{III.3}$$

We denote this algorithm *TVD-C*, where *TVD* stands for Time-Varying Densities, and *C* stands for Centralized, as will be discussed in subsequent sections. Note that if a CVT is perfectly achieved, then the proportional term does not contribute anything to the update law, and the result in Theorem III.1 still applies. Although we do not present a formal proof of convergence for Equation III.3, it has been verified in simulation and experiments that a CVT can be achieved using this controller, even in the case when the robots are far from a CVT condition.

The second issue with Equation III.2 is the presence of the term $\partial c/\partial p$. Even though this might look innocent, this term is rather complicated, due to the fact that

$$c_i(p,t) = \int_{V_i(p)} q\phi(q,t)dq \Big/ \int_{V_i(p)} \phi(q,t)dq,$$

which depends on p in the boundary of the area over which the two integrals are taken. As a result, Leibniz rule must be exercised.[1]

IV. DISTRIBUTED APPROXIMATIONS

Given a Voronoi partition, we will denote the boundary between two cells by ∂V_{ij}. In the planar case, this boundary is either empty (Voronoi cells do not intersect), a single point (Voronoi cells intersect at a single vertex) or a line (Voronoi cells share a face). The two Voronoi cells are said to be adjacent if they share a face, and we denote the set of cells adjacent to cell i by N_{V_i}.

[1]The resulting line and area integrals are practically computed using numerical approximations (e.g., Riemann sums, Gaussian quadrature) in the robotic implementation and for human-generated densities, the partial $\partial\phi/\partial t$ can be approximated by a finite difference scheme.

Now, suppose that $i \notin N_{V_j}$. This means either $\partial V_{i,j}$ is empty or consists of a singleton. This moreover implies that any integrals over $\partial V_{i,j}$ are zero, and Leibniz rule tells us that these integrals are what define $\frac{\partial c_i}{\partial p_j}$, from which we can conclude that $\frac{\partial c_i}{\partial p_j} = 0$. As such we have the following result

Lemma IV.1. $i \notin N_{V_j} \Rightarrow \frac{\partial c_i}{\partial p_j} = 0.$

A direct consequence of Lemma IV.1 is that $\partial c/\partial p$ encodes adjacency information. And, both algorithms in Equation II.7 and III.1 are distributed in this manner, i.e., the update rule for $\dot{p}_i$ only depends on p_j if $j \in N_{V_i}$. This, however, is not the case with Equation III.3 since even though $\partial c/\partial p$ has the right sparsity structure, $(I - \partial c/\partial p)^{-1}$ does not. In fact, the inverse renders the resulting matrix dense and all sparsity structure is lost. The purpose of this section is thus twofold: 1) to develop a distributed approximation to III.3 and 2) to overcome the restrictions associated with ϕ for the inverse in III.3 to exist.

Lemma IV.2 (Neumann series)**.** *Let A be a square matrix. If $\lim_{k\to\infty} A^k = 0$, then $I - A$ is invertible and*

$$(I-A)^{-1} = I + A + A^2 + A^3 + \dots$$

Moreover, for a $m \times m$ square matrix A, $\lim_{k\to\infty} A^k = 0$ if and only if $|\lambda_i| < 1$ for all $i = 1,2,\cdots,m$, where λ_i are the eigenvalues of A. As such, let λ_{max} denote the eigenvalue with the largest magnitude of the matrix $\partial c/\partial p$. Using the Neumann series, we can express $(I - \partial c/\partial p)^{-1}$ as

$$\left(I - \frac{\partial c}{\partial p}\right)^{-1} = I + \frac{\partial c}{\partial p} + \left(\frac{\partial c}{\partial p}\right)^2 + \dots$$

as long as $|\lambda_{max}| < 1$.

Now, if we insist on only letting $\dot{p}_i$ depend on $p_j,\ j \in N_{V_i}$, (as well as p_i itself) we can truncate the series after just two entries

$$\left(I - \frac{\partial c}{\partial p}\right)^{-1} \approx I + \frac{\partial c}{\partial p},$$

which gives the update law (modified from III.3),

$$\dot{p} = \left(I + \frac{\partial c}{\partial p}\right)\left(-\kappa(p-c) + \frac{\partial c}{\partial t}\right),$$

or at the level of the individual robots

TVD-D_1 :

$$\dot{p}_i = \frac{\partial c_i}{\partial t} - \kappa(p_i - c_i) + \sum_{j\in N_{V_i}} \frac{\partial c_i}{\partial p_j}\left(\frac{\partial c_j}{\partial t} - \kappa(p_j - c_j)\right), \tag{IV.1}$$

where the label denotes Time-Varying-Density, Decentralized with 1-hop adjacency information.

It should be noted that Equation IV.1 is always well-defined (as long as ϕ is continuously differentiable). In other words, even if the Neumann series is not convergent or if the inverse does not exist, the entries in IV.1 are well-defined.

One can now investigate what happens when higher order terms are kept in the Neumann series. For this, we let $dist(i,j)$ denote the distance between cells i and j.[2] And, as $\partial c/\partial p$ is

[2]Formally speaking, $dist(i,j)$ is the edge distance, or number of edges in the shortest path, between i and j in the Delaunay graph induced by the Voronoi tessellation.

TABLE I: Total costs under different $TVD\text{-}D_k$

Algorithm	Total cost for ϕ_1	Total cost for ϕ_2
$TVD\text{-}D_0$	316.7	37.3
$TVD\text{-}D_1$	309.8	35.9
$TVD\text{-}D_2$	308.2	35.8
$TVD\text{-}D_4$	307.1	35.8
$TVD\text{-}D_{10}$	306.5	35.8
$TVD\text{-}C$	306.4	35.8

a (block) adjacency matrix, we have that

$$\left[(\partial c/\partial p)^k\right]_{ij} \neq 0 \;\Rightarrow\; dist(i,j) \leq k,\; k = 0,1,2,\ldots$$

where $[\cdot]_{ij}$ denotes the block corresponding to cell c_i and robot position p_j.

The k-hop version of $TVD-D_1$ thus becomes

$TVD\text{-}D_k$:

$$\dot{p} = \sum_{\ell=0}^{k} \left(\frac{\partial c}{\partial p}\right)^{\ell} \left(-\kappa(p-c) + \frac{\partial c}{\partial t}\right), \qquad \text{(IV.2)}$$

V. IMPLEMENTATION

A. Experimental Results

In the previous section, a family of distributed algorithms, $TVD\text{-}D_k$, $k = 0,1,\ldots$, were developed as approximations to $TVD\text{-}C$, which, in turn, was presented as an alternative to the two algorithms dubbed *Lloyd* (Equation II.7) and *Cortes* (Equation III.1). In this section, we implement these different algorithms on a team of mobile robots, both in simulation and on Khepera III differential-drive mobile robots.

Two different density functions were considered

$$\phi_1(q,t) = e^{-\left((q_x - 2\sin\left(\frac{t}{\tau}\right)\right)^2 + \left(\frac{q_y}{4}\right)^2\right)}$$
$$\phi_2(q,t) = e^{-\left((q_x - 2\cos\left(\frac{t}{\tau}\right)\right)^2 + \left(q_y - 2\sin\left(\frac{t}{\tau}\right)\right)^2\right)}.$$

The choice of density function is arbitrary, but these were selected for their difference in spatial symmetry and translational velocities. The "time constant" τ was taken to be $\tau = 5$, and as a sanity-check, a number of simulations were performed using $TVD\text{-}D_1$ from different initial conditions.

Moreover, different versions of $TVD\text{-}D_k$, were simulated for ϕ_1 and ϕ_2 with the total cost being

$$\int_0^{T_f} H(p(t),t)dt.$$

The costs are summarized in Table I. And, as can be seen, the cost does indeed decrease slightly as more terms are kept in the Neumann series. However, the difference between the different cases is not particularly dramatic beyond the $k = 0$ to $k = 1$ case, i.e., when no information is used about neighboring robot positions and when only adjacent neighbors are taken into account. Similarly, the price of anarchy, i.e., the difference between $TVD\text{-}D_1$ and $TVD\text{-}C$ is marginal.

Moreover, a comparison was made to *Lloyd* and to *Cortes*, using ϕ_1 and ϕ_2 as well as three additional time-varying density functions – one of which (ϕ_5) was generated using human inputs, as discussed subsequently. In all of these cases, the robots were initialized to the same positions to mitigate an inherently problematic comparison, since these algorithms are chasing local (as opposed to global) minimizers to the locational cost. The performance metric used was the total locational cost, the same as in Table I. These findings are summarized in Table II.

TABLE II: Coverage performance comparison.

	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5
$TVD\text{-}D_1$	309.8	35.0	36.5	35.9	100.2
$TVD\text{-}C$	306.4	34.3	34.3	33.7	98.9
Cortes	319.5	38.4	N/A	37.5	101.7
Lloyd	324.6	40.1	52.6	38.7	103.6

In all cases (except ϕ_3) *Cortes* did indeed perform better than *Lloyd*, which is not surprising since *Lloyd* is designed for static density functions. However, under density function ϕ_3, the assumptions behind *Cortes* were violated. Moreover, $TVD\text{-}C$ and $TVD\text{-}D_1$ both outperformed *Cortes* and *Lloyd* in all five cases. In all of those cases, $TVD\text{-}C$ performed best, as can be expected. Among the decentralized algorithms, $TVD\text{-}D_1$ was the overall most effective algorithm since it is always well-posed, allows for a distributed implementation, and performs better than the previously proposed algorithms.

$TVD\text{-}D_1$ was implemented on a team of mobile robots. The ROS (Robot Operating System, version Diamondback) framework running on Ubuntu (version 11.04) machine with Intel dual core CPU 2.13GHz, 4GB memory was used to implement the algorithm and send control signals to individual robots over a wireless router. Five Khepera III robots from K-team were used as the team of mobile robots for the experiment. The Khepera III robots each have a 600MHz ARM processor with 128Mb RAM, embedded Linux, differential drive wheels, and a wireless card for communication over a wireless router. Ten OptiTrack S250e motion capture cameras were used to provide position and orientation data for the robots, which were used to provide the information required for the algorithm and the computation of the Voronoi partitions.

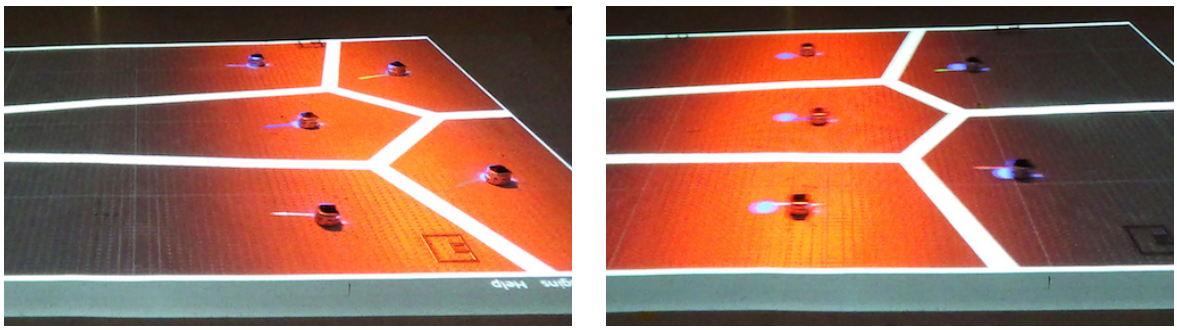
As the Khepera III mobile robots are differential-drive robots, they can be modeled as unicycles,

$$\dot{x}_i = v_i \cos\theta_i, \qquad \dot{y}_i = v_i \sin\theta_i, \qquad \dot{\theta}_i = \omega_i,$$

where (x_i, y_i) is the position of robot i, θ_i its heading, and v_i, ω_i are the translational and angular velocities. In contrast to this, the coverage algorithm provides desired motions in terms of $\dot{p}_i$ and we map these onto v_i, ω_i through $v_i = \|\dot{p}_i\|$ and $\omega = [-\sin\theta_i, \cos\theta_i]\dot{p}_i / \|\dot{p}_i\|$. The result is shown in Fig. 1.

B. Human Generated Density Functions

Although achieving coverage over time-varying density functions is useful in its own right, our motivation stems from human-operated teams of mobile robots. The idea is that the density functions serve as input modalities to the system, and that a human operator should be able to manipulate these densities.



(a) (b)

Fig. 1: $TVD\text{-}D_1$ is deployed on a team of five mobile robots for density ϕ_1. An overhead projector is visualizing pertinent information: the thick lines delineate the Voronoi cells, whose centroids are shown as the bright dots. The arrows show each robot's desired direction of motion. (See `http://youtu.be/fu5LrlBcu9Y`.)

The way densities were generated was by utilizing a touch-sensing input device (a tablet), whereby the user can provide the desired configuration by drawing the regions of interest over the domain. To reduce the amount of information required to describe the drawn density, while maintaining a continuously differentiable density ϕ, the density can be approximated with parametric densities. We opted for a Gaussian Mixture Model (GMM) to approximate the density for the purpose of proof-of-concept. The GMM density was made time-varying by translating the mean value of every centroid in the GMM by a human-generated continuous function, which "dragged" the density around.

VI. Conclusion and Future Directions

This paper presents a novel coverage algorithm that can handle time-varying density functions as well as lend itself to a distributed implementation, and experimental results demonstrate the viability of the proposed approach. The main idea is to combine a proportional term driving the robots to the centroid of their Voronoi cells with a controller tracking the time-varying evolution.

It should be noted, however, that in practice, input saturations, modeling errors associated with the single integrator dynamics, and aggressively varying density functions make the robots temporarily deviate from their centroids on occasions. This fact seems to indicate that this paper should be thought of as a first step towards handling time-varying density functions and that more robust and responsive methods can be developed as future endeavors. The generation of effective density functions from human inputs is another issue that will be pursued in the future, as well as the introduction of obstacles and non-convex areas of interest.

References

[1] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks: Variations on a theme," in *Mediterranean Conference on Control and Automation*, Lisbon, Portugal, July 2002, Electronic Proceedings.

[2] ——, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[3] J. Cortés and F. Bullo, "Coordination and Geometric Optimization via Distributed Dynamical Systems," *SIAM Journal on Control and Optimization*, vol. 44, no. 5, pp. 1543–1574, Oct. 2005.

[4] L. C. A. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. S. Pereira, "Simultaneous Coverage and Tracking (SCAT) of Moving Targets with Robot Networks," in *Algorithmic Foundation of Robotics VIII*, ser. Springer Tracts in Advanced Robotics, G. S. Chirikjian, H. Choset, M. Morales, and T. Murphey, Eds. Springer Berlin Heidelberg, 2010, vol. 57, pp. 85–99.

[5] D. Haumann, V. Willert, and K. D. Listmann, "DisCoverage: From Coverage to Distributed Multi-Robot Exploration," in *Proceedings of the 4th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, vol. 4, no. 1, Koblenz, Germany, September 2013, pp. 328–335.

[6] S. Martinez, J. Cortes, and F. Bullo, "Motion Coordination with Distributed Information," *Control Systems Magazine, IEEE*, vol. 27, no. 4, pp. 75–88, 2007.

[7] A. Macwan, G. Nejat, and B. Benhabib, "Target-Motion Prediction for Robotic Search and Rescue in Wilderness Environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 41, no. 5, pp. 1287–1298, 2011.

[8] A. Ghaffarkhah, Y. Yan, and Y. Mostofi, "Dynamic coverage of time-varying environments using a mobile robot – A communication-aware perspective," in *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*, 2011, pp. 1297–1302.

[9] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, Dec. 1999.

[10] M. Iri, K. Murota, and T. Ohya, "A fast Voronoi-diagram algorithm with applications to geographical optimization problems," in *System Modelling and Optimization*, ser. Lecture Notes in Control and Information Sciences, P. Thoft-Christensen, Ed. Springer Berlin Heidelberg, 1984, vol. 59, pp. 273–288.

[11] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the Lloyd Algorithm for Computing Centroidal Voronoi Tessellations," *SIAM Journal on Numerical Analysis*, vol. 44, no. 1, pp. 102–119, Jan. 2006.

[12] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, "On centroidal voronoi tessellation – energy smoothness and fast computation," *ACM Transactions on Graphics*, vol. 28, no. 4, pp. 1–17, Sep. 2009.

[13] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[14] Q. Du and M. Emelianenko, "Acceleration schemes for computing centroidal Voronoi tessellations," *Numerical Linear Algebra with Applications*, vol. 13, no. 2-3, pp. 173–192, 2006.