Eyes in the Sky: Decentralized Control for the Deployment of Robotic Camera Networks

Citation: Schwager, Mac et al. "Eyes in the Sky: Decentralized Control for the Deployment of Robotic Camera Networks." Proceedings of the IEEE 99.9 (2011): 1541–1561.

As Published: http://dx.doi.org/10.1109/jproc.2011.2158377

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Persistent URL: http://hdl.handle.net/1721.1/72505

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Eyes in the Sky: Decentralized Control for the Deployment of Robotic Camera Networks

Mac Schwager, *Member IEEE*, Brian J. Julian, *Student Member IEEE,*
Michael Angermann, *Member IEEE*, and Daniela Rus, *Fellow IEEE*

***Abstract*—This paper presents a decentralized control strategy for positioning and orienting multiple robotic cameras to collectively monitor an environment. The cameras may have various degrees of mobility from six degrees of freedom, to one degree of freedom. The control strategy is proven to locally minimize a novel metric representing information loss over the environment. It can accommodate groups of cameras with heterogeneous degrees of mobility (e.g. some that only translate and some that only rotate), and is adaptive to robotic cameras being added or deleted from the group, and to changing environmental conditions. The robotic cameras share information for their controllers over a wireless network using a specially designed networking algorithm. The control strategy is demonstrated in repeated experiments with three flying quadrotor robots indoors, and with five flying quadrotor robots outdoors. Simulation results for more complex scenarios are also presented.**

***Index Terms*—Multirobot systems; distributed control; networked control systems; wireless sensor networks; mobile ad hoc networks; unmanned aerial vehicles; distributed algorithms; nonlinear control systems**

Manuscript received May 29, 2010; revised November 2, 2010; accepted May 7, 2011. This work was supported in part by the ARO MURI SWARMS grant number W911NF-05-1-0219, the ONR MURI SMARTS grant number N000140911051, NSF grant numbers IIS-0426838 and EFRI-0735953, the MAST project, MIT Lincoln Laboratory, and the Boeing Company. This work is sponsored by the Department of the Air Force under Air Force contract number FA8721-05-C-0002. The opinions, interpretations, recommendations, and conclusions are those of the authors and are not necessarily endorsed by the United States Government.

M. Schwager is with the Department of Mechanical Engineering, Boston University, Boston, MA 02215. He was previously with the Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT, Cambridge, MA 02139, USA. (e-mail: schwager@mit.edu).

B. J. Julian is with the Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT, Cambridge, MA 02139, USA, and MIT Lincoln Laboratory, Lexington, MA 02420, USA (e-mail: bjulian@csail.mit.edu).

M. Angermann is with the German Aerospace Institute (DLR), Institute of Communication and Navigation, Wessling, Germany (e-mail: michael.angermann@dlr.de).

D. Rus is with the Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT, Cambridge, MA 02139, USA (e-mail: rus@csail.mit.edu).

I. Introduction

Camera networks are all around us. They are used to monitor retail stores, catch speeding drivers, collect military intelligence, and gather scientific data. Soon autonomous aircraft with cameras will be routinely surveilling our cities, our neighborhoods, and our wildlife areas. This technology promises far reaching benefits for the study and understanding of large-scale complex systems, both natural and man made. However, before we can realize the potential of camera networks, we must address an important technical question: how should a group of cameras be positioned in order to maintain the best view of an environment? In this paper we provide a comprehensive method of controlling groups of robotic cameras in a decentralized way to guarantee visual coverage of a given environment.

We consider the problem of providing visual coverage with maximal resolution using a group of robots with cameras. The robot group can be heterogeneous in that some cameras may be fixed to aerial or ground robots, while others may be able to pan and tilt in place. Our goal is to control the robots in a decentralized fashion to autonomously position and orient their cameras so that the union of their fields of view achieves visual coverage of a given planar environment at a maximal resolution. We propose a controller with stability and convergence guarantees based on a gradient descent strategy to drive the robots to such a configuration.

Existing camera surveillance systems often use cameras that are mounted to actuated mechanisms for adjusting the orientations of the cameras. Furthermore, it is becoming increasingly common to mount cameras to autonomous ground and air vehicles, for example the iRobot PackBot or the Northrup Gruman Global Hawk. In this paper we consider each camera along with its positioning mechanism, be it a rotating mounting or an autonomous vehicle, as a “robot,” and assume that there is a wireless network in place to facilitate communication among the robots. We formulate a decentralized control strategy for the cameras to position themselves in an automated and adaptive way in order to maintain the best view of the environment. Our controller is demonstrated with a group of automated helicopter robots, known as quadrotors, fitted with downward facing cameras. We present results with groups of three quadrotors in an indoor environment and five quadrotors in an outdoor environment.

The control strategy we describe is useful for robustly collecting visual data over large scale environments either for security or scientific applications. We envision the algorithm as being used in support of a higher-level computer vision task, such as object recognition or tracking. That is, we address the problem of how to best position the robots given that the images from their cameras will be used by some computer vision algorithm. For example, the controller could be used to drive groups of autonomous underwater or aerial vehicles to do mosaicing [1], or to produce photometric stereo from multiple camera views [2]. This might be applied to imaging underwater or land-based archaeological sites or geological formations, environments of ecological interest such as coral reefs or forests, regions that are inaccessible to humans such as disaster sites or war zones, or any other large scale environment of interest. Our algorithm could also be used by autonomous flying robots to do surveillance [3], target tracking [4]–[6], or to provide real-time localization and mapping to aid in the navigation of people or vehicles on the ground [7].

Our approach is motivated by an information content principle: minimum information per pixel. Using information per pixel as a metric allows for the incorporation of physical, geometric, and optical parameters to give a cost function that represents how well a group of cameras covers an environment. We obtain a control law by taking the negative gradient of this cost function. The controller is proved to converge to a local minimum of the cost function using Lyapunov techniques.[1]

The controller is naturally adaptive to the deletion or addition of cameras to the group, and to a changing environment, and will work with a broad class of environment geometries, including ones with nonconvexities, and ones with multiple disconnected regions. The controller is also decentralized in that robots only exchange information with other robots whose fields of view intersect with its own, and are not aware of the size nor the composition of the whole group. In the case that two robots with intersecting fields of view are not in direct communication with one another, we describe an efficient networking algorithm for state propagation so that information can be routed between these robots. Finally, the controller also accommodates heterogeneous groups in that different robots in the group may be able to move their cameras in different ways. For example some cameras may only translate while others may only pan and tilt. This provides insights and tools for studying the tradeoffs between re-positioning a camera versus rotating it in place.

[1]The camera coverage task is necessarily nonconvex, as proved in [8], and thus gradient controllers can only achieve locally optimal configurations.

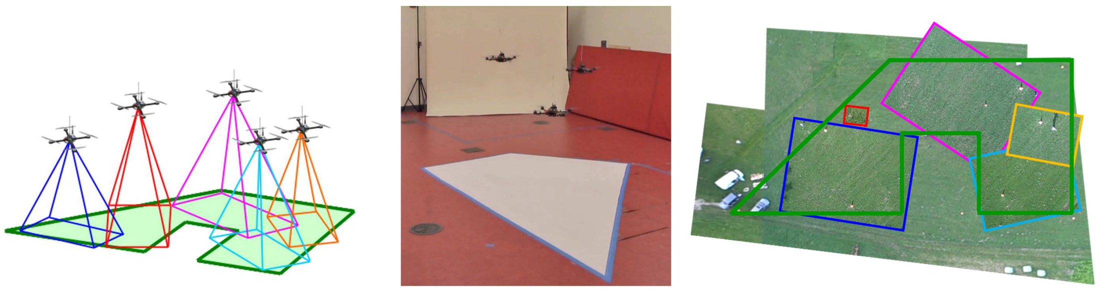
The main contributions of this work are as follows.

1) We propose the minimum information per pixel principle as a cost function for camera placement.
2) We use the cost function to design a provably-stable controller to deploy multiple robots with fixed downward facing cameras to locally optimal positions in a distributed fashion.
3) We generalize the problem formulation to design a provably-stable controller for heterogeneous systems whose cameras have as many as six degrees of freedom.
4) We introduce a practical algorithm for enabling communication of the necessary position information around the wireless mesh network.
5) We present simulation results for several scenarios including ones with heterogeneous groups of robots.
6) We implement the controller on quadrotor robots with fixed downward facing cameras, and provide results from multiple experiments for three quadrotor robots in an indoor environment and five quadrotor robots outdoors.

A. Related Work

Much of the work in this paper is inspired by a recent body of research concerning the optimal deployment of robots for providing sensor coverage of an environment. Cortés et al. [9] introduced a stable distributed controller for sensor coverage based on ideas from the optimal facility placement literature [10]. This approach involves a Voronoi partition of the environment and has seen several extensions, for example to covering nonconvex environments [11]–[13], to learning some aspect of the environment on-line [14], and to incorporate collision avoidance [12]. One recent extension described in [15], Figure 14, proposed an algorithm for the placement of hovering sensors, similar to our scenario.

Our method in this paper is related to this body of work in that we propose a cost function and obtain a distributed controller by taking its gradient. However, the cost function we propose is different from previous ones in that it does not involve a Voronoi partition.



(a) Concept (b) Experiment (c) Photo Mosaic

Fig. 1. This figure shows the main concept and an example implementation of our decentralized coverage controller. Our control strategy positions multiple flying robots with cameras to cover an environment in a decentralized way, as in the schematic in 1(a). Experiments were carried out in an indoor environment with three robots and in an outdoor environment with five robots, as shown in 1(b). The resulting images form the aerial cameras can be stitched together to produce a large scale surveillance image, as in 1(c).

To the contrary, it relies on the fields of view of multiple cameras to overlap with one another. Another distinction from previous works is that the agents we consider move in a space that is different from the one they cover. Previous coverage scenarios have considered agents constrained to move in the environment that they cover, which leads to a requirement that the environment must be convex. This requirement can be overcome with more sophisticated algorithms, but it has been shown in the literature to be a non-trivial limitation [11]–[13]. In contrast, we consider agents moving in a space in $\mathbb{R}^3$, covering an arbitrary lower dimensional environment $Q \subset \mathbb{R}^2$, which eliminates the need for the environment Q to be convex. Indeed, it need not even be connected. It must only be Lebesgue measurable (since the robots will calculate integrals over it), which is quite a broad specification.

There have also been other algorithms for camera placement, for example a probabilistic approach for general sensor deployment based on the Cramér-Rao bound was proposed in [16], and an application of the idea for cameras was given in [17]. In [18] the authors choose to focus on positioning downward facing cameras, as opposed to arbitrarily oriented cameras. Many geometrical aspects of the problem are significantly simplified in this setting. More generally, several other works have considered cooperative control with flying robots and UAV's. For an excellent review of cooperative UAV control please see [19], or [20] and [21] for two recent examples.

The remainder of the paper is organized as follows. In Section II we formulate the problem of optimally covering an environment with cameras. In Section III we introduce the decentralized controller and analyze its convergence and stability properties for a homogeneous multi-robot system with fixed downward pointing cameras. In Section IV we show an extension to rotating cameras, beginning with one rotational degree of freedom, then generalizing to three rotational degrees of freedom, and finally to heterogeneous groups made up of robots with various degrees of freedom. Section V presents simulation results for the cases of a homogeneous system with fixed cameras with three rotational degrees of freedom, a homogeneous system with cameras with three translational degrees of freedom, and a heterogeneous system with rotating and translating cameras. Section VI proposes an mesh networking algorithm for propagating the information required by the controller to all of the robots. Finally, Section VII describes hardware experiments with three quadrotor robots indoors and five quadrotor robots outdoors, and conclusions are given in Section VIII. Preliminary versions of some of the results in this paper have appeared in [22]–[25].

II. Optimal Camera Placement

We motivate our approach with an informal justification of a cost function, then develop the problem formally for the single camera case followed by the multi-camera case. We desire to cover a bounded environment, $Q \subset \mathbb{R}^2$, with a number of cameras. We assume Q is planar, without topography, to avoid the complications of changing elevation or occlusions. Let $p_i \in \mathcal{P}$ represent the state of camera i, where the state space, $\mathcal{P}$, will be characterized later. We want to control n cameras in a distributed fashion such that their placement minimizes the aggregate information per camera pixel over the environment,

$$\min_{(p_1,\ldots,p_n)\in\mathcal{P}^n} \int_Q \frac{\text{info}}{\text{pixel}}\, dq.$$

This metric makes sense because the pixel is the fundamental information capturing unit of the camera. Consider the patch of the environment that is exposed to a single pixel, as represented by the red circle in Figure 2. The information in that patch is reduced by the camera to a low-dimensional representation (i.e. mean color and brightness over the patch). Therefore, the less information content the image patch contains, the less information will be lost in its low dimensional representation by the pixel. Furthermore, we want to minimize the accumulated information loss due to pixelation over the whole environment Q, hence the integral. In the next two sections we will formalize the notion of information per pixel.

A. Single Camera

We develop the cost function for a single camera before generalizing to multiple cameras. It is convenient to consider the information per pixel as the product of two functions, $f : \mathcal{P} \times Q \mapsto (0, \infty]$, which gives the area in the environment seen by one pixel (the "area per pixel" function), and $\phi : Q \mapsto (0, \infty)$ which gives the information per area in the environment. The form of $f(p_i, q)$ will be derived from the optics of the camera and geometry of the environment. The function $\phi(q)$ is a positive weighting of importance over Q and should be specified beforehand (it can also be learned from sensor data, as in [14]). For instance, if all points in the environment are equally important, $\phi(q)$ should be constant over Q. If some known area in Q requires more resolution, the value of $\phi(q)$ should be larger in that area than elsewhere in Q. This gives the cost function

$$\min_p \int_Q f(p, q)\phi(q)\, dq, \tag{1}$$

which is of a general form common in the locational optimization and optimal sensor deployment literature [10], [26]. We will introduce significant changes to this basic form with the addition of multiple cameras.

The state of the camera, p, consists of all parameters associated with the camera that effect the area per pixel function, $f(p, q)$. In a general setting one might consider the camera's position in $\mathbb{R}^3$ and its angular orientation (which can be represented by a matrix in $SO(3)$), as well as camera specific parameters such as a zoom factor in $(0, \infty)$, thus leading to an optimization in a rather complicated state-space, $\mathcal{P} = \mathbb{R}^3 \times SO(3) \times (0, \infty)$, for only one camera. For this reason, we first consider the special case in which the camera is downward facing (hovering over Q). This case is of particular interest in many applications involving surveillance with autonomous vehicles, as described in Section I. We will first consider a camera with a circular field of view because this considerably simplifies the geometry and allows us to neglect all rotational degrees of freedom. In Section IV-A we will consider a downward facing camera with a rectangular field of view, so that one rotational degree of freedom becomes relevant, followed by the case with three rotational degrees of freedom and a rectangular field of view in Section IV-B.

We define the field of view, $\mathcal{B}$, to be the intersection of the cone whose vertex is the focal point of the camera lens with the subspace that contains the environment, as shown in Figure 2. In this case $\mathcal{P} = \mathbb{R}^3$, and the state-space in which we do optimization is considerably simplified from that of the unconstrained camera. Decompose the camera position as $p = [c^T, z]^T$, with $c \in \mathbb{R}^2$ the lateral position of the focal point of the camera, and $z \in \mathbb{R}$ the height of the focal point of the camera over Q. We have

$$\mathcal{B} = \left\{ q \mid \frac{\|q - c\|}{z} \leq \tan\theta \right\} \tag{2}$$

where θ is the half-angle of view of the camera.

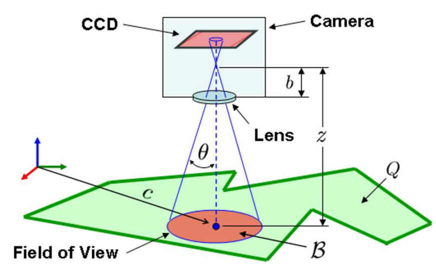


Fig. 2. The camera optics and the geometry of the environment are shown in this figure.

To find the area per pixel function, $f(p, q)$, consider the geometry in Figure 2. Let b be the focal length of the lens. Inside $\mathcal{B}$, the area/pixel is equal to the inverse of the area magnification factor (which is defined from classical optics [27] to be $b^2/(b-z)^2$) times the area of one pixel. Define a to be the area of one pixel divided by the square of the focal length of the lens. We have,

$$f(p, q) = \begin{cases} a(b - z)^2 & \text{for} \quad q \in \mathcal{B} \\ \infty & \text{otherwise.} \end{cases} \tag{3}$$

Outside of the field of view there are no pixels, therefore the area per pixel is infinite. The cost function in (1) takes on an infinite value if any area (of non-zero measure) of Q is outside of the field of view. However we know there exists a $p \in \mathcal{P}$ such that the cost is finite since Q is bounded (given c and θ, there exist $z \in \mathbb{R}$ such that $Q \subset \mathcal{B}$). Therefore, we can write the equivalent constrained optimization problem

$$\begin{aligned} \min_p \textstyle\int_Q a(b-z)^2\phi(q)\,dq, \\ \text{subject to} \quad Q \subset \mathcal{B}. \end{aligned} \tag{4}$$

One can see in this simple scenario that the optimal solution is for p to be such that the field of view is the smallest ball that contains Q. However, with multiple cameras, the problem becomes more challenging.

B. Multiple Cameras

To find optimal positions for multiple cameras, we have to determine how to account for the area of overlap of the images of the cameras, as shown in Figure 3. Intuitively, an area of Q that is being observed by two

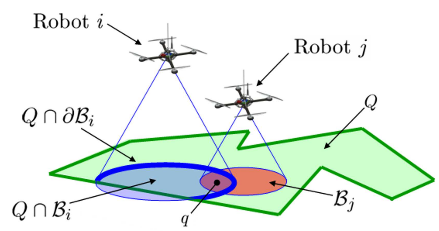


Fig. 3. This figure shows the relevant quantities involved in characterizing the intersecting fields of view of two cameras.

different cameras is better covered than if it were being observed by only one camera, but it is not *twice* as well covered. Consider a point q that appears in the image of n different cameras. The number of pixels per area at that point is the sum of the pixels per area for each camera. Therefore the area per pixel at that point is given by the *inverse* of the sum of the *inverse* of the area per pixel for each camera, or

$$\frac{\text{area}}{\text{pixel}} = \Big(\sum_{i=1}^{n} f(p_i,q)^{-1}\Big)^{-1},$$

where p_i is the position of the ith camera. We emphasize that it is the *pixels per area* that sum because of the multiple cameras, not the *area per pixel* because, in the overlap region, multiple pixels are observing the same area. Therefore the inverse of the sum of inverses is unavoidable. Incidentally, this is the same form one would use to combine the variances of multiple noisy measurements when doing Bayesian sensor fusion [8].

Finally, we introduce a prior area per pixel, $w \in (0,\infty)$. The interpretation of the prior is that there is some pre-existing photograph of the environment (e.g. an initial reconnaissance photograph), from which we can get a base-line area per pixel measurement. This is compatible with the rest of our scenario, since we will assume that the robots have knowledge of the geometry of the environment Q, and some notion of information content over it, $\phi(q)$. This pre-existing information can be arbitrarily vague (w can be arbitrarily large) but it must exist. The prior also has the benefit of making the cost function finite for all robot positions. It is combined with the camera sensors as if it were another camera to get

$$\frac{\text{area}}{\text{pixel}} = \Big(\sum_{i=1}^{n} f(p_i,q)^{-1} + w^{-1}\Big)^{-1},$$

Let $\mathcal{N}_q$ be the set of indices of cameras for which $f(p_i,q)$ is bounded, $\mathcal{N}_q = \{i \mid q \in \mathcal{B}_i\}$. We can now write the area per pixel function as

$$h_{\mathcal{N}_q}(p_1,\ldots,p_n,q) = \Big(\sum_{i\in\mathcal{N}_q} f(p_i,q)^{-1} + w^{-1}\Big)^{-1}. \tag{5}$$

to give the cost function

$$\mathcal{H}(p_1,\ldots,p_n) = \int_Q h_{\mathcal{N}_q}(p_1,\ldots,p_n,q)\phi(q)\,dq. \tag{6}$$

We will often refer to $h_{\mathcal{N}_q}$ and $\mathcal{H}$ without their arguments. Now we can pose the multi-camera optimization problem,

$$\min_{(p_1,\ldots,p_n)\in\mathcal{P}^n} \mathcal{H}. \tag{7}$$

The cost function (6) is of a general form valid for any area per pixel function $f(p_i,q)$, and for any camera state space $\mathcal{P}$ (including cameras that have rotational degrees of freedom). Notice also that $\mathcal{H} > 0$ for all $(p_1,\ldots,p_n)$. We proceed with the special case of downward facing cameras, where $\mathcal{P} = \mathbb{R}^3$ and $f(p_i,q)$ is given by (3).

III. Decentralized Control

We will take the gradient of (6) and find that it is distributed among the robots in the sense that for a robot to compute its component of the gradient, it only needs to know the state of the other robots whose fields of view intersect with its own. This will lead to a decentralized gradient-based controller. We will use the notation $\mathcal{N}_q\backslash\{i\}$ to mean the set of all indices in $\mathcal{N}_q$, except for i.

Theorem 1 (Gradient Component)**.** *The gradient of the cost function $\mathcal{H}(p_1,\ldots,p_n)$ with respect to a robot's position p_i, using the area per pixel function in (3) is given by*

$$\frac{\partial\mathcal{H}}{\partial c_i} = \int_{Q\cap\partial\mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\backslash\{i\}})\frac{(q-c_i)}{\|q-c_i\|}\phi(q)\,dq, \tag{8}$$

and

$$\frac{\partial \mathcal{H}}{\partial z_i} = \int_{Q\cap\partial\mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\setminus\{i\}})\phi(q)\tan\theta\, dq - \int_{Q\cap\mathcal{B}_i} \frac{2h_{\mathcal{N}_q}^2}{a(b-z_i)^3}\phi(q)\, dq. \qquad (9)$$

Proof. *Please refer to the appendix for a proof.*

We propose to use a gradient control law in which every robot follows the negative of its own gradient component,

$$u_i = -k\partial\mathcal{H}/\partial p_i, \qquad (10)$$

where u_i is the control input for robot i and $k \in (0,\infty)$ is a control gain. Assuming integrator dynamics for the robots,

$$\dot{p}_i = u_i, \qquad (11)$$

we can prove the convergence of this controller to locally minimize the aggregate information per area.

Theorem 2 (Convergence and Stability)**.** *For a network of n robots with the dynamics in (11), using the controller in (10),*

i) $\lim_{t\to\infty} \frac{\partial\mathcal{H}}{\partial p_i} = 0 \quad \forall i \in \{1,\ldots,n\}$,
ii) *An equilibrium $(p_1^*,\ldots,p_n^*)$, defined by $\frac{\partial\mathcal{H}}{\partial p_i}\ |_{p_i=p_i^*} = 0 \quad \forall i \in \{1,\ldots,n\}$, is Lyapunov stable if and only if it is a local minimum of $\mathcal{H}$.*

Proof (Convergence and Stability)**.** *The proof of statement i) is an application of LaSalle's invariance principle ([28], [26] Theorem 1.17 [2]). Let $\mathcal{H}(p_1,\ldots,p_n)$ be a Lyapunov-type function candidate. The closed-loop dynamics $\dot{p}_i = -\partial\mathcal{H}/\partial p_i$ do not depend on time, and $\partial\mathcal{H}/\partial p_i$ is a continuous function of p_j for all j, therefore the dynamics are locally Lipschitz, and $\mathcal{H}$ is continuously differentiable. Taking the time derivative of $\mathcal{H}$ along the trajectories of the system gives*

$$\dot{\mathcal{H}} = \sum_{i=1}^{n} \frac{\partial\mathcal{H}}{\partial p_i}^T \dot{p}_i = -\sum_{i=1}^{n} \frac{\partial\mathcal{H}}{\partial p_i}^T \frac{\partial\mathcal{H}}{\partial p_i} \leq 0. \qquad (12)$$

Next we show that all evolutions of the system are bounded. To see this, consider a robot at p_i such that $Q\cap\mathcal{B}_i = \emptyset$. Then $\dot{p}_i = 0$ for all time (if the field of view leaves Q, the robot stops for all time), so $c_i(t)$ is bounded. Given $Q\cap\mathcal{B}_i \neq \emptyset$, $\mathcal{H}$ is radially unbounded (i.e. coercive) in z_i, therefore $\dot{\mathcal{H}} \leq 0$ implies that z_i is bounded for all time. Finally, consider the set of all $(p_1,\ldots,p_n)$ for which $\dot{\mathcal{H}} = 0$. This is itself an invariant set, since $\dot{\mathcal{H}} = 0$ implies $\partial\mathcal{H}/\partial p_i = \dot{p}_i = 0$ for all i. Therefore, all conditions of LaSalle's principle are satisfied and the trajectories of the system converge to this invariant set.

There may exist configurations at which $\frac{\partial\mathcal{H}}{\partial p_i} = 0\ \forall i$ that are saddle points, local maxima, or local minima of $\mathcal{H}$. Statement ii) says that only the local minima *of $\mathcal{H}$ are stable equilibria. A proof of this intuitively obvious fact about gradient systems can be found in [29], Chapter 9, Section 4.*

[2]In this application, the invariance principle requires (1) autonomous, locally Lipschitz dynamics, (2) a non-increasing, continuously differentiable Lyapunov function, (3) all evolutions of the system remain bounded.

Remark 1 (Intuition)**.** *The single integral for the lateral component (8) causes the robot to move to increase the amount of the environment in its field of view, while also moving away from other robots j whose field of view overlaps with its own. The vertical component (9) has two integrals with competing tendencies. The first integral causes the robot to move up to bring more of the environment into its field of view, while the second integral causes it to move down to get a better look at the environment already in its field of view.*

Remark 2 (Requirements)**.** *Both the lateral (8) and vertical (9) components can be computed by robot i with knowledge of 1) its own position, p_i, 2) the environment, Q, 3) the information per area function, $\phi(q)$, and 4) the positions of all other robots whose fields of view intersect with its own (which can be found by communication or sensing).*

Remark 3 (Network Requirements)**.** *The requirement that a robot can communicate with all other robots whose fields' of view intersect with its own describes a minimal network graph for our controller to be feasible. In particular, we require the network to be at least a proximity graph in which all agents i are connected to all other agents $j \in \mathcal{N}_i$, where $\mathcal{N}_i = \{j \mid Q\cap\mathcal{B}_i\cap\mathcal{B}_j \neq \emptyset, i \neq j\}$. To compute the controller over a network that is a subgraph of the required proximity graph, a robot needs an algorithm for maintaining estimates of the states of the robots with whom it is not in direct communication. Such an algorithm is discussed in Section VI. In the case that the network becomes disconnected, the separate connected sub-groups will tend to come together as each sub-group tries to entirely cover the environment (being unaware of the other sub-groups). In the case that they do not reconnect, all connected sub-groups will separately cover the environment on their own.*

Remark 4 (Adaptivity)**.** *The controller is adaptive in the sense that it will stably reconfigure if any number of robots fail. It will also work with nonconvex environments, Q, including disconnected ones. In the case of a disconnected environment, the robots may (or may not, depending on the specific scenario) split into a number of sub-groups that are not in communication with one another. The controller can also track changing environments, Q, and changing information per area functions, $\phi(q)$, provided these quantities change slowly enough. This is not addressed by the theorem, but has been shown to be the case in simulation studies.*

Remark 5 (Control Gains and Robustness)**.** *The proportional control gain, k, adjusts the aggressiveness of the controller. In a discretized implementation one should set this gain low enough to provide robustness to discretization errors and noise in the system. The prior area per pixel, w, adjusts how much of the area Q will remain uncovered in the final configuration. It should be chosen to be as large as possible, but as with k, should be small enough to provide robustness to discretization errors and noise in the system.*

Remark 6 (Obstacles and Collisions)**.** *The controller does not explicitly take into account collisions with obstacles or with other robots. The natural tendency of the controller is for robots to push away from one another, though this does not give a definite guarantee, and analytical results to this effect would be difficult to obtain. In a practical setting, this controller would have to be combined with an obstacle and collision avoidance controller in either a hybrid or blended control architecture to prevent collisions. In the 30 experimental trials described in this paper, no collision avoidance component was used, and collisions were not a problem, except for a single instance in which a faulty gyro sensor resulted in a midair collision of two quadrotors.*

This controller can be implemented in a discretized setting as Algorithm 1. In general, the integrals in the controller must be computed using a discretized approximation. Let $\widehat{Q \cap \partial\mathcal{B}_i}$ and $\widehat{Q \cap \mathcal{B}_i}$ be the discretized sets of gird points representing the sets $Q \cap \partial\mathcal{B}_i$ and $Q \cap \mathcal{B}_i$, respectively. Let Δq be the length of an arc segment for the discretized set $\widehat{Q \cap \partial\mathcal{B}_i}$, and the area of a grid square for the discretized set $\widehat{Q \cap \mathcal{B}_i}$. A simple algorithm that approximates (10) is then given in Algorithm 1.

To determine the computational complexity of this algorithm, let us assume that there are m points in both sets $\widehat{Q \cap \partial\mathcal{B}_i}$ and $\widehat{Q \cap \mathcal{B}_i}$. We can now calculate the time

Algorithm 1 Discretized Controller

Require: Robot i knows its position p_i, the extent environment Q, and the information per area function $\phi(q)$.

Require: Robot i can communicate with all robots j whose field of view intersects with its own.

loop

Communicate with neighbors to get p_j

Compute and move to

$$c_i(t+\Delta t) = c_i(t) \\ -k \textstyle\sum_{q \in \widehat{Q \cap \partial\mathcal{B}_i}} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q \setminus \{i\}}) \frac{(q-c_i)}{\|q-c_i\|} \phi(q) \Delta q$$

Compute and move to

$$z_i(t+\Delta t) = z_i(t) \\ -k \textstyle\sum_{q \in \widehat{Q \cap \partial\mathcal{B}_i}} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q \setminus \{i\}}) \phi(q) \tan\theta \Delta q \\ +k \textstyle\sum_{q \in \widehat{Q \cap \mathcal{B}_i}} \frac{2h_{\mathcal{N}_q}^2}{a(b-z_i)^3} \phi(q) \Delta q$$

end loop

complexity as

$$T(n,m) \leq \sum_{j=1}^{m} (O(1) + \sum_{k=1}^{n} O(1)) + \\ \sum_{j=1}^{m} (O(1) + \sum_{k=1}^{n} O(1) + \sum_{k=1}^{n-1} O(1)) \in O(nm).$$

When calculating the controller for all robots on a centralized processor (as was done for the simulations in Section V), the time complexity becomes $T(n,m) \in O(n^2 m)$.

IV. Extension to Rotating Cameras

Until this point we have assumed that the camera's field of view, $\mathcal{B}_i$, is a circle, and that the camera is fixed in a downward pointing position. Of course, actual cameras have a rectangular CCD array, and therefore a rectangular field of view. This means that the rotational orientation of the camera with respect to the ground must also be controlled. Furthermore, one may want to mount the camera on gimbals to control pan and tilt angles. This would introduce another two rotational degrees of freedom that must be controlled. In this section we revisit the gradient in Theorem 1 and calculate it first for a rectangular field of view and one degree of rotational freedom, and then consider a rectangular field of view with the full six degrees of freedom. Finally, we consider the case of heterogeneous groups made up of cameras with different degrees of freedom.

A. Rectangular Field of View

Let the state space of $p_i = [c_i^T \quad z_i \quad \psi_i]^T$ be $\mathcal{P} = \mathbb{R}^3 \times \mathbb{S}$, where ψ_i is the yaw angle. The rotation matrix in $SO(2)$ associated with ψ_i is given by

$$R(\psi_i) = \begin{bmatrix} \cos\psi_i & \sin\psi_i \\ -\sin\psi_i & \cos\psi_i \end{bmatrix}, \tag{13}$$

where $R(\psi_i)q$ rotates a vector q expressed in the global coordinate frame, to a coordinate frame aligned with the axes of the rectangular field of view. As is true for all rotation matrices, $R(\psi_i)$ is orthogonal, meaning $R(\psi_i)^T = R(\psi_i)^{-1}$. Using this matrix, define the field of view of robot i to be

$$\mathcal{B}_i = \{q \mid |R(\psi_i)(q - c_i)| \leq z_i \tan\theta\}, \tag{14}$$

where $\theta = [\theta_1 \quad \theta_2]^T$ is a vector with two angles which are the half-view angles associated with two perpendicular edges of the rectangle, as shown in Figure 4, and the $\leq$ symbol applies element-wise (all elements in the vector must satisfy $\leq$). We have to break up the boundary of the rectangle into each of its four edges. Let l_k be the kth edge, and define four outward-facing normal vectors n_k, one associated with each edge, where $n_1 = [1 \quad 0]^T$, $n_2 = [0 \quad 1]^T$, $n_3 = [-1 \quad 0]$, and $n_4 = [0 \quad -1]$.

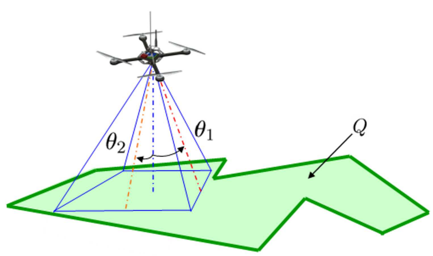


Fig. 4. The geometry of a camera with a rectangular field of view is shown in this figure.

The cost function, $\mathcal{H}(p_1, \ldots, p_n)$, is the same as for the circular case, as is the area per pixel function $f(p_i, q)$.

Theorem 3 (Rectangular Gradient)**.** *The gradient of the cost function $\mathcal{H}(p_1, \ldots, p_n)$ with respect to a robot's position p_i using the area per pixel function in (3) and the rectangular field of view in (14) is given by*

$$\frac{\partial\mathcal{H}}{\partial c_i} = \sum_{k=1}^{4} \int_{Q\cap l_{k\,i}} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\setminus\{i\}}) R(\psi_i)^T n_{k\,i} \phi(q)\, dq, \tag{15}$$

$$\frac{\partial\mathcal{H}}{\partial z_i} = \sum_{k=1}^{4} \int_{Q\cap l_{k\,i}} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\setminus\{i\}}) \tan\theta^T n_{k\,i} \phi(q)\, dq - \int_{Q\cap\mathcal{B}_i} \frac{2h_{\mathcal{N}_q}^2}{a(b - z_i)^3} \phi(q)\, dq, \tag{16}$$

and

$$\frac{\partial\mathcal{H}}{\partial\psi_i} = \sum_{k=1}^{4} \int_{Q\cap l_{k\,i}} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\setminus\{i\}}) \cdot (q - c_i)^T R(\psi_i + \pi/2)^T n_{k\,i} \phi(q)\, dq. \tag{17}$$

Proof. *Please see the Appendix for a proof.*

The terms in the gradient have interpretations similar to the ones for the circular field of view. The lateral component (15) has one integral which tends to make the robots move away from neighbors with intersecting fields of view, while moving to put its entire field of view inside of the environment Q. The vertical component (16) comprises two integrals. The first causes the robot to go up to take in a larger view, while the second causes it to go down to get a better view of what it already sees. The angular component (17) rotate the robot to get more of its field of view into the environment, while also rotating away from other robots whose field of view intersects its own. Computation of the gradient component for the rectangular field of view is of the same complexity as the circular case, and carries the same constraint on the communication topology.

B. Incorporating Pan and Tilt Angles

In the previous section we extended the controller to the case of four degrees of freedom: three positional degrees and one angular degree. In this section we complete the extension to the most general case, six degrees of freedom, by including pan and tilt angles. The full six degrees of freedom can be realized with a camera mounted on double gimbals to a hovering robot. The robot's position and yaw angle account for the position and rotation angle of the camera while the gimbals control pan and tilt angles of the camera.

The full freedom of motion complicates the geometry of the field of view considerably. The field of view is a trapezoid in this case, the lengths of whose sides and the angles between them depend nonlinearly upon the six degrees of freedom of the camera. One can most easily visualize the geometry by considering a rectangular pyramid emanating from the focal point of the lens of the camera toward the environment. We will call this the field of view pyramid, or just the pyramid. This pyramid intersects with the plane of the environment to create the field of view of the camera. The plane of the environment can be oriented arbitrarily with respect to the pyramid, creating a trapezoidal field of view (assuming the pan and tilt angle are within certain limits so that all sides of the pyramid intersect the plane). Please refer to Figure 5 for a schematic of the geometry involved.

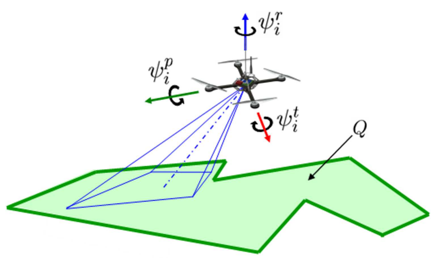


Fig. 5. This figure shows the distortion of the field of view into a trapezoid due to pan and tilt angles of the camera. This considerably complicates the geometry for cameras with six degrees of freedom.

We follow a similar procedure as for the rectangular case, analyzing the geometry to obtain the geometric constraints describing the field of view, and differentiating the constraints to obtain the gradient controller. Stability can be proved in the same way as before by appealing to Theorem 2 about the convergence and stability of the gradient system.

To formulate the geometry involved, we will introduce a system of coordinate frames and 3-dimensional rotations to describe the state of the camera. Let the state of camera i be given by $p_i = [x_i \quad y_i \quad z_i \quad \psi_i^r \quad \psi_i^p \quad \psi_i^t]^T$, and the state space is $\mathcal{P} = \mathbb{R}^3 \times \mathbb{S}^1 \times [-\frac{\pi}{2}+\theta_1, \frac{\pi}{2}-\theta_1] \times [-\frac{\pi}{2}+\theta_2, \frac{\pi}{2}-\theta_2]$, where θ_1 and θ_2 are the two half angles of view as defined above. The angle ψ_i^r is the rotation (or yaw) angle, which is positive when the field of view spins clockwise, ψ_i^p is the pan (or roll) angle, which is positive when the field of view sweeps to the left, and ψ_i^t is the tilt (or pitch) angle, which is positive when the field of view sweeps upward. The ranges for pan and tilt are limited to the angles over which the field of view is bounded. We also introduce $\rho_i = [x_i \quad y_i \quad z_i]^T$ to denote the position of the focal point of the camera. We represent the orientation of the camera by the angles that have to be controlled in the gimbals mechanism (similarly to Euler angles), however we will also deal with their associated rotation matrix in $SO(3)$ to represent the field of view trapezoid.

Consider two coordinate frames: the Camera Fixed frame of robot i (CF_i) and the Global Fixed frame (GF), which is the same for all robots. The CF_i is fixed to the camera, centered at the focal point, with the z-axis pointing through the lens and the y-axis pointing out the right side of the camera. The GF is centered at a fixed origin on the ground, with the z-axis pointing upward normal to the ground. To express vectors in either the CF_i or GF frames conveniently, we first formulate three rotation matrices in $SO(3)$, each realizing a rotation through a rotational angle, as

$$R_i^r = \begin{bmatrix} \cos\psi_i^r & \sin\psi_i^r & 0 \\ -\sin\psi_i^r & \cos\psi_i^r & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_i^p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi_i^p & \sin\psi_i^p \\ 0 & -\sin\psi_i^p & \cos\psi_i^p \end{bmatrix},$$

$$\text{and} \quad R_i^t = \begin{bmatrix} \cos\psi_i^t & 0 & -\sin\psi_i^t \\ 0 & 1 & 0 \\ \sin\psi_i^t & 0 & \cos\psi_i^t \end{bmatrix}.$$

To take a point, x, in the GF and express it in the CF_i we first translate the vector by ρ_i, the position of the focal point in the GF, then rotate the vector about the z axis by $\pi/2$ and flip it about the x-axis by π using the matrix

$$R^f = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{20}$$

and, finally, rotate it through ψ_i^r, ψ_i^p, and ψ_i^t in sequence using the rotation matrices in (18). This gives the transformation $R_i(\psi_i^t, \psi_i^p, \psi_i^r)(x - \rho_i)$, which is an element of the special Euclidean group $SE(3) = \mathbb{R}^3 \times SO(3)$, where

$$R_i(\psi_i^t, \psi_i^p, \psi_i^r) = R_i^t R_i^p R_i^r R^f. \tag{21}$$

We will henceforth drop the angle arguments from R_i to be concise. Likewise, we can take a point y in the CF_i frame and express it in the GF frame with the inverse transformation in $SE(3)$ as $R_i^T y + \rho_i$. We will use these transforms to write the constraints that describe the trapezoidal field of view of the camera.

Consider the four outward facing unit normal vectors of the faces of the field of view pyramid. Denote them in the CF_i frame in counter-clockwise order, starting from the right-hand face of the pyramid as

$$m_{1i} = [0 \quad \cos\theta_1 \quad -\sin\theta_1]^T, \quad m_{2i} = [\cos\theta_2 \quad 0 \quad -\sin\theta_2]^T,$$

$$m_{3i} = [0 \quad -\cos\theta_1 \quad -\sin\theta_1], \quad \text{and} \quad m_{4i} = [-\cos\theta_2 \quad 0 \quad -\sin\theta_2]$$

Let the kth leg of the trapezoidal field of view be called l_k as before. The vector from the focal point, ρ_i, to a point in the leg, q, is perpendicular to the normal of the kth pyramid face, therefore

$$m_{ki}^T R_i (I_{3,2} q - \rho_i) = 0, \tag{24}$$

where $I_{i,j}$ is the $i \times j$ identity matrix. We defined q to be in $\mathbb{R}^2$ (embedded in the ground plane), so we must express it in $\mathbb{R}^3$, appending a zero z coordinate with $I_{3,2}q$. Points on or to the left of l_k (when looking in the counter-clockwise direction), satisfy ${m_k}_i^T R_i(I_{3,2}q - \rho_i) \leq 0$. Therefore the field of view can be described by

$$\mathcal{B}_i = \{q \mid {m_k}_i^T R_i(I_{3,2}q - \rho_i) \leq 0 \quad k = 1, 2, 3, 4\}. \quad (25)$$

It is also useful to explicitly state the vertices of the field of view. Let v_{ki} be the vertex between the legs l_{k-1} and l_k (where l_{k-1} is understood to be l_4 for $k = 1$). Then the vertex must satisfy (24) for both k and $k-1$, which gives $[m_{k-1\,i} \quad m_{ki}]^T R_i(I_{3,2}v_{ki} - \rho_i) = 0$. Solving for the vertex v_{ki} gives

$$v_{ki} = ([m_{k-1\,i} \quad m_{ki}]^T R_i I_{3,2})^{-1}[m_{k-1\,i} \quad m_{ki}]^T R_i \rho_i. \quad (26)$$

Now that we have defined the field of view, we must revisit the area per pixel function, $f(p_i, q)$. Previously, we implicitly approximated the distance from the point in the environment, q, to the camera focal point, ρ_i, to be z_i, which is a fair approximation if the camera remains pointed at the ground. Now, however, we must account for the fact that points on one side of the field of view may be significantly closer to the focal point than points on the other side because of the tilt and pan of the camera. For this reason we re-define $f(p_i, q)$ to be

$$f(p_i, q) = \begin{cases} a(b - \|I_{3,2}q - \rho_i\|)^2 & \text{for} \quad q \in \mathcal{B}_i \\ \infty & \text{otherwise.} \end{cases} \quad (27)$$

The cost function, $\mathcal{H}(p_1, \ldots, p_n)$, is the same as for the circular and rectangular cases. The difference is only in the specification of the field of view, $\mathcal{B}_i$, which is given by (25), and the new area per pixel function specified by (27). To derive the gradient controller, however, we must differentiate the constraint equation (24) as before. We relegate the details of this differentiation to the appendix, and show the result in the form of a theorem.

Theorem 4 (Six Degree of Freedom Gradient)**.** *The gradient of the cost function $\mathcal{H}(p_1, \ldots, p_n)$ with respect to a camera's six degree of freedom state $p_i = [x_i \quad y_i \quad z_i \quad \psi_i^r \quad \psi_i^p \quad \psi_i^t]^T$ using the area per pixel function in (27) and the trapezoidal field of view defined by (25), is given by*

$$\frac{\partial \mathcal{H}}{\partial \rho_i} = \sum_{k=1}^{4} \int_{Q \cap l_{ki}} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q \setminus \{i\}}) \frac{R_i^T m_{ki}}{\|I_{2,3}R_i^T m_{ki}\|} \phi(q)\, dq + \int_{Q \cap \mathcal{B}_i} \frac{2h_{\mathcal{N}_q}^2}{a(b - \|I_{3,2}q - \rho_i\|)^3} \frac{(I_{3,2}q - \rho_i)}{\|I_{3,2}q - \rho_i\|} \phi(q)\, dq, \quad (28)$$

$$\frac{\partial \mathcal{H}}{\partial \psi_i^s} = \sum_{k=1}^{4} \int_{Q \cap l_k} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q \setminus \{i\}}) \frac{{m_k}_i^T \frac{\partial R_i}{\partial \psi_i^s}(\rho_i - I_{3,2}q)}{\|I_{2,3}R_i^T m_k\|} \phi(q)\, dq \quad s \in \{r, p, t\}, \quad (29)$$

where

$$\frac{\partial R_i}{\partial \psi_i^r} = R_i^t R_i^p \begin{bmatrix} -\sin\psi_i^r & \cos\psi_i^r & 0 \\ -\cos\psi_i^r & -\sin\psi_i^r & 0 \\ 0 & 0 & 0 \end{bmatrix} R^f, \quad \frac{\partial R_i}{\partial \psi_i^p} = R_i^t \begin{bmatrix} 0 & 0 & \cdots \\ 0 & -\sin & \cdots \\ 0 & -\cos & \cdots \end{bmatrix}$$

$$\text{and} \quad \frac{\partial R_i}{\partial \psi_i^t} = \begin{bmatrix} -\sin\psi_i^t & \cdots \\ 0 & \cdots \\ \cos\psi_i^t & \cdots \end{bmatrix}$$

Proof. *Please see the Appendix for a proof.*

The controller in (10) can now be used with the gradient above to produce a controller for the full six degree of freedom case.

C. Heterogeneous Groups

The gradient control scheme that we propose can be directly applied to heterogeneous groups of robots. If a robot is restricted so that some of its rotational or translational variables are constant, one can apply the controller in (10) to whatever components in the gradient in Theorem 4 are controllable. For example, consider a two robot group in which one robot can only translate and one robot can only rotate. Then the state space associated with the translating robot is $\mathcal{P}_1 = \mathbb{R}^3$, that for the rotating robot is $\mathcal{P}_2 = \mathbb{S}^1 \times [-\frac{\pi}{2} + \theta_1, \frac{\pi}{2} - \theta_1] \times [-\frac{\pi}{2} + \theta_2, \frac{\pi}{2} - \theta_2]$, and the state space for the whole system is $\mathcal{P}_1 \times \mathcal{P}_2$. The relevant optimization for this robot group becomes

$$\min_{(p_1, p_2) \in \mathcal{P}_1 \times \mathcal{P}_2} \mathcal{H}_{\text{het}}(p_1, p_2). \quad (31)$$

The gradient of $\mathcal{H}_{\text{het}}$ above is the same as the gradient of $\mathcal{H}$, except that $\mathcal{H}_{\text{het}}$ is only a function of variables $p_1 = [x_1 \quad y_1 \quad z_1]$ and $p_2 = [\psi_{r2} \quad \psi_{p2} \quad \psi_{t2}]$, so its gradient only has elements with respect to these six variables. This applies in general to situations in which any robot has degrees of freedom which are a subset of the six possible degrees of freedom. The convergence and stability results in Theorem 2 still hold since the controller is still a gradient controller, and if $\mathcal{H}$ is continuously differentiable, then $\mathcal{H}_{\text{het}}$ is also.

One can also readily extend to the case in which robots' states are constrained to lie on a manifold in their state space, that is, if their state variables are constrained to maintain some relationship with respect to one another. The gradient can be calculated in a straightforward manner using the chain rule. For example, suppose we have control over x_i, but y_i is constrained such that $y_i = g(x_i)$. Then the gradient of the constrained cost function $\mathcal{H}_{\text{cnstr}}$ is simply found from the unconstrained cost function by

$$\frac{\partial \mathcal{H}_{\text{cnstr}}(x_i)}{\partial x_i} = \frac{\partial \mathcal{H}}{\partial x_i} + \frac{\partial \mathcal{H}}{\partial y_i}\frac{\partial g}{\partial x_i}. \tag{32}$$

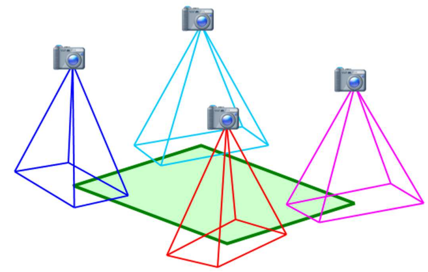
As long as the constraint g is differentiable, with a locally Lipschitz derivative, the convergence and stability in Theorem 2 are ensured. Other kinds of constraints (for example those written as $g(x_i, y_i) = 0$) can be handled in a similar way. In the next section we demonstrate the proposed control scheme for the three cases of a homogeneous group of robots with fixed cameras, a homogeneous group of robots with cameras that can pan and tilt, and a heterogeneous group of robots.

V. Simulations

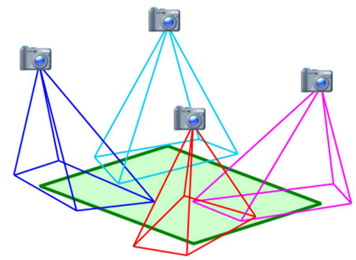
We conducted numerical simulations to demonstrate the performance of the algorithm in various situations. The cameras were simulated in a Matlab environment and controlled using Algorithm 1 on a centralized processor. The camera parameters were set to $a = 10^{-6}$, $b = 10$, $\theta_1 = 35$ deg, $\theta_2 = 20$ deg, which are typical for commercially available hand held digital cameras. The control gains were set to $\phi = 1$, $w = 2^{16}$, and $k = 10^{-6}[1 \quad 1 \quad .1 \quad 10^{-9} \quad 10^{-9} \quad 10^{-9}]^T$. We will show the results from three representative simulation scenarios here.

The first simulation, shown in Figure 6, models a scenario in which there are four surveillance cameras in a square room, one in each upper corner. The cameras can rotate about each of their three rotational axes, but cannot translate. The relevant controller then uses only (29) in 1. The cameras begin pointing downward (Figure 6(a)), then they rotate their fields of view into the square environment (Figure 6(b)), and finally arrange themselves so that each covers a different patch of the environment, while allowing for some overlap (Figure 6(c)). The decreasing value of the cost function $\mathcal{H}$ is shown in Figure 6(d). The final value of the function is very small compared to the initial value, but it is not zero. Indeed the cost function is always greater than zero, as can be seen from the definition of $\mathcal{H}$ in (7). The function appears to decrease jaggedly because of the discretized integral computation in Algorithm 1.

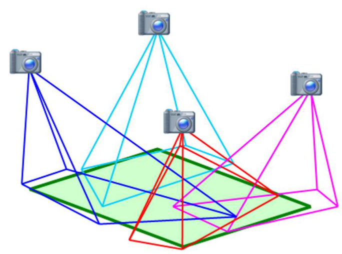
The second simulation is of five flying robots with downward facing cameras. The robots (and their cameras) have three translational degrees of freedom and can rotate about their yaw axis. The controller equations from Algorithm 1 were computed with the gradient in (15), (16), and (17). The environment in this case is nonconvex. This scenario is similar to our outdoor experiments performed with quadrotor robots as described



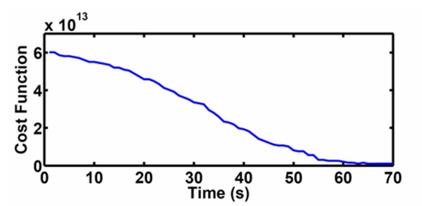
(a) Initial Config.



(b) Middle Config.



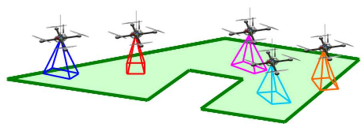
(c) Final Config.



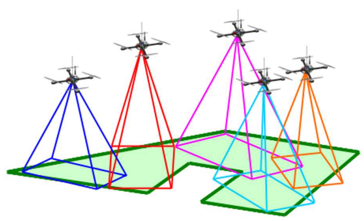
(d) Cost Function

Fig. 6. Results of a simulation with four cameras fixed in the corners of a square room are shown. The cameras can rotate about all three rotational axes. The camera icon marks the camera positions and the pyramids represent the fields of view of the cameras. The initial, middle, and final configurations are shown in 6(a), 6(b), and 6(c), respectively. The decreasing value of the aggregate information per pixel function, $\mathcal{H}$, is shown in 6(d). The jaggedness of the curve is due to the discretized integral approximation.

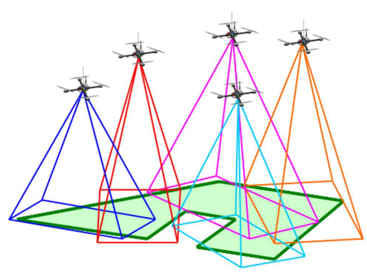
in Section VII-B. Figure 6 shows the results of a typical simulation. The robots start in an arbitrary configuration and spread out and up so that their fields of view cover the environment. As in the previous simulation, the cost function appears jagged because of the discretized integral computation in Algorithm 1.



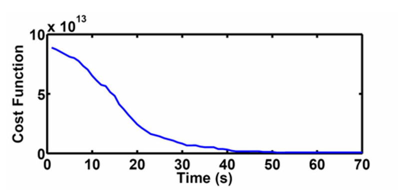
(a) Initial Config.



(b) Middle Config.



(c) Final Config.



(d) Cost Function

Fig. 7. Results of a simulation with five cameras on flying robots (indicated by the quadrotor icons) are shown over a nonconvex environment. The cameras can translate in all three axis and can rotate about the yaw axis. The pyramids represent the fields of view of the cameras. The initial, middle, and final configurations are shown in 7(a), 7(b), and 7(c), respectively. The decreasing value of the aggregate information per pixel function, $\mathcal{H}$, is shown in 7(d). The jaggedness of the curve is due to the discretized integral approximation.

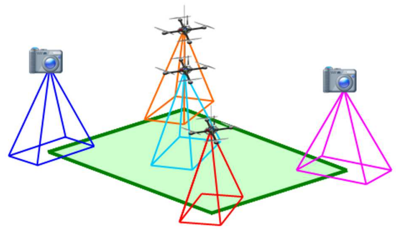
The final simulation scenario is of two fixed cameras in opposite corners of a room, similarly to the first scenario, along with three cameras mounted to flying robots and gimbals to enable motion in all six degrees of freedom. This is, therefore, an example of a heterogeneous group, as described in Section IV-C. The figures show the flying cameras spreading out and up over the environment, while the fixed cameras sweep their fields of view into the environment. The cameras eventually settle in a configuration in which the team covers the environment. Figure 8(d) shows the decreasing cost of the group, and is again jagged due to the discretized integral in the computation of the controller.

VI. Propagating States Over the Network

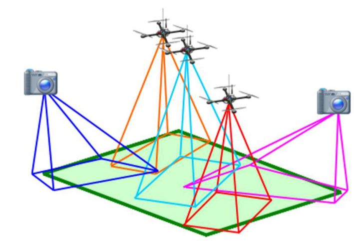
In this section we describe a networking algorithm to support the camera coverage controller described above. The algorithm facilitates the efficient propagation of robot state information around the network by weighting the frequency with which robot j's state information is sent to robot i by how relevant robot j's state is to robot i's controller.

As discussed in Remark 3, the camera coverage controller requires the communication of state information between robots with overlapping fields of view. Unfortunately, there is no practical way to guarantee that robots with overlapping fields of view will be in direct communication with one another. Many of the envisioned applications for our control algorithm require the robot team to spread over large-scale domains where distances between robots can become larger than their transmission ranges. Furthermore, transmission ranges depend on complicated factors beyond inter-robot distance, such as environment geometry, channel interference, or atmospheric conditions. Therefore, to implement the proposed controller, we require a practical multi-hop networking algorithm to distribute state information over the entire system.

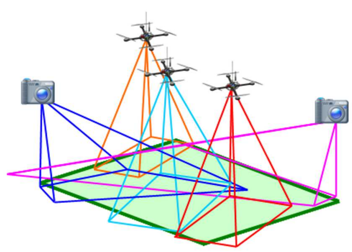
Existing mobile ad hoc networks typically use sophisticated routing schemes to pass data packets around the network. Due to the mobile nature of such networks, these schemes consume a significant amount of communication capacity for maintaining knowledge about network topology. They also are not efficient (in terms of time, bandwidth, and power) for our application because they do not prioritize state information based on its relevance to the controller. Instead, we here propose an algorithm tailored for our application that is more likely to broadcast state information of robots that are near by than of those that are far away. The algorithm ensures that the state information most likely to be used by a robot's controller is also most likely to be up-to-date. This location-based multi-hop algorithm increases propagation rates of state estimates in local neighborhoods (i.e. robots that are likely to have overlapping fields of view),



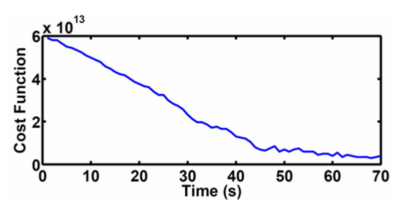
(a) Initial Config.



(b) Middle Config.



(c) Final Config.



(d) Cost Function

Fig. 8. Results of a simulation with two cameras fixed in the corners of a square room and three cameras mounted on flying robots are shown. The fixed cameras (denoted by the camera icons) can rotate about all three rotational axes while the flying cameras (denoted by the quadrotor icons) have the full six degrees of freedom. The initial, middle, and final configurations are shown in 8(a), 8(b), and 8(c), respectively. The decreasing value of the aggregate information per pixel function, $\mathcal{H}$, is shown in 8(d). The jaggedness of the curve is due to the discretized integral approximation.

while also being efficient in terms of bandwidth and computational complexity. We refer the reader to [23] for a detailed description and analysis of the algorithm.

A. *Importance-Based Broadcasting*

In this section we formalize the idea of maintaining state estimates over a network and propose a means of prioritizing state information based upon proximity. Consider n robots, each of which knows its current state, $p_i(t) \in \mathcal{P}$, by some means of measurement (e.g. GPS or visual localization). We propose that each robot maintains a list of state estimates, $[p_1(t_{i1}), \dots, p_n(t_{in})]$, where t_{ij} denotes a time stamp at which robot i's estimate of robot j's state was valid. We have that $t_{ij} \leq t$ and $t_{ii} = t$.

For simplicity, we use Time Division Multiple Access (TDMA)[3] to divide the data stream into time slots of length γ. During a time slot, one assigned robot is allowed to broadcast over the shared channel. The length γ is measured by the number of state estimates (along with their time stamps) that can be broadcast in the time slot. For example, with a slot of length $\gamma = 5$ a robot can transmit 5 state estimates. The robots broadcast one after the other in a predetermined order. One complete broadcast cycle is referred to as a frame. The length of a frame is proportional to $n\gamma$.

One naive strategy, called simple flooding, is to assign a time slot length equal to the number of robots, $\gamma = n$, so that each robot can broadcast its entire list of state estimates. Although simple to implement, this strategy is not scalable for a large number of robots since increasing the number of robots in the system will *quadratically* decrease the frame rate (i.e. the rate the team can cycle through all time slots). This highlights the inherent tradeoff between the amount of information that can be broadcast, and the currency of that information. Our algorithm seeks to balance that trade-off.

Consider a function $g : \mathcal{P} \times \mathcal{P} \mapsto (0, \infty]$, called the importance function, that weights how important it is for robot i to have a current state estimate of robot j, defined as

$$g_{ij}(t) = \|p_i(t) - p_j(t_{ij})\|^{-1}. \tag{33}$$

A robot should consider its own state estimate to be the most important to broadcast. This is reflected in the model since g_{ii} is infinite. We use the importance function in (33) to develop a deterministic algorithm. For a given time slot, this algorithm selects which state estimates a robot will broadcast. We first describe a probabilistic approach to help motivate the final algorithm.

[3]The proposed strategy is not limited to only TDMA; many other channel access methods are appropriate (e.g. FDMA or CDMA).

B. Probabilistic Algorithm

Consider a robot that needs to select l state estimates to broadcast during its time slot. We provided motivation in Section VI-A that some selections are more important than others. However, the robot should *not* systematically select the state estimates associated with the highest importance; doing so can prevent estimates from fully dispersing throughout the system. Instead, we propose that the probability of robot i selecting the state estimate of robot j is

$$P^{ij}_{\mathcal{M}_i}(t) = \frac{g_{ij}(t)}{\sum_{k \in \mathcal{M}_i} g_{ik}(t)}, \quad j \in \mathcal{M}_i \tag{34}$$

where $\mathcal{M}_i$ is the set of robot indices associated with selectable estimates.

Prior to the first selection for a given time slot, $\mathcal{M}_i$ is the set of all robot indices. From the full set the robot always selects its own state since it has infinite importance. The robot then removes its index from $\mathcal{M}_i$. Since (34) is a valid probability mass function, the robot can simply choose the next state estimate at random from the corresponding probability distribution, then remove the corresponding index from $\mathcal{M}_i$. This means estimates of closer robots are more likely to be chosen than ones that are farther away. By repeating this process, the entire time slot of length γ can be filled in a straightforward, probabilistic manner.

C. Deterministic Algorithm

The probabilistic method above is not suitable in practice because consecutive selections of a particular robot index can be separated by an undesirably long period of time, especially for distant robots. By developing a location-based deterministic algorithm, we can increase the average rate at which all state estimates of a given time stamp will propagate throughout a team. In the deterministic case, propagation time is bounded above by the longest path taken among the estimates. No such bound exists in the probabilistic case, resulting in a positively skewed distribution of propagation times and a larger mean. We propose that each robot maintains a list of counters, $[c_{i1}, \ldots, c_{in}]$, which are initially set to a value of one. Using the probability mass function in (34), each counter represents the probability that the corresponding index has *not* been selected. Consider a robot's first selection, which will always be its own index. The probability, $P^{ii}_{\mathcal{M}_i}(t)$, of selecting index i is equal to one, while all other probabilities, $P^{ij}_{\mathcal{M}_i}(t)$ subject to $j \neq i$, are equal to zero. This implies that the counter c_{ii} is multiplied by $[1 - P^{ii}_{\mathcal{M}_i}(t)] = 0$, or a zero probability of not being selected, while all other counters, c_{ij}, are multiplied $[1 - P^{ij}_{\mathcal{M}_i}(t)] = 1$, or a probability of one. By selecting the index with the lowest counter value, we are deterministically guiding our method to behave according to the probability distribution described by (34). The selected index (in this case i) is removed from the set $\mathcal{M}_i$, and its corresponding counter (c_{ii}) is reset to a value of one. This process is iteratively applied to completely fill a time slot with γ state estimates, with counters maintaining their values between frames. The complete deterministic strategy is given in Algorithm 2.

Algorithm 2 Deterministic Method for Selecting State Estimates

n is the number of robots in the system and l is the time slot length.
Require: Robot i knows its state $p_i(t)$ and the state estimate of other robots $p_j(t_{ij})$.
Require: Robot i knows its running counter $[c_{i1}, \ldots, c_{in}]$.
$\mathcal{M}_i \leftarrow \{1, \ldots, n\}$
for 1 to γ **do**
  $P^{ij}_{\mathcal{M}_i}(t) \leftarrow \frac{g_{ij}(t)}{\sum_{k \in \mathcal{M}_i} g_{ik}(t)}, \;\; \forall j \in \mathcal{M}_i$
  $c_{ij} \leftarrow c_{ij}[1 - P^{ij}_{\mathcal{M}_i}(t)], \;\; \forall j \in \mathcal{M}_i$
  $k \leftarrow \arg\max_{k \in \mathcal{M}_i}(c_{ik})$
  $\mathcal{M}_i \leftarrow \mathcal{M}_i \backslash \{k\}$
  $c_{ik} \leftarrow 1$
end for
return $\{1, \ldots, n\} \backslash \mathcal{M}_i$

VII. Experiments

To demonstrate the performance of our distributed control algorithm, we conducted both indoor and outdoor experiments using multiple Ascending Technologies (AscTec) Hummingbird quadrotor flying robots. The pitch, roll, and yaw angles of the robots were stabilized at 1 kHz using the on-board commercial controller developed for the platform as described in [30]. We developed a custom microprocessor module, described in [25], to run the coverage algorithm in this paper. This high-level controller calculated position waypoints for the robot's closed-loop position controller at 1 Hz. We found that a 1 Hz update rate for the waypoint commands is sufficiently slow compared to the settling time of the position controller that the robot's dynamics are well approximated by the integrator dynamics in (11).

A. Optimal Coverage of an Indoor Environment

Our indoor experiments were performed at the Computer Science and Artificial Intelligence Lab (CSAIL) at

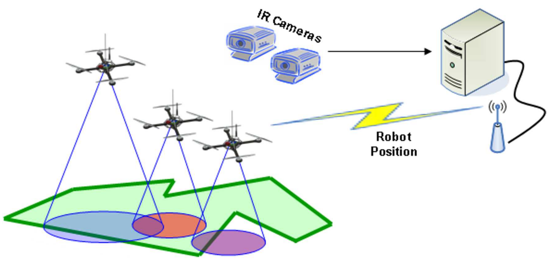


Fig. 9. This figure shows the experimental setup. The robots positions were captured with a Vicon motion capture system. The robots used their position information to run the coverage algorithm in a distributed fashion.

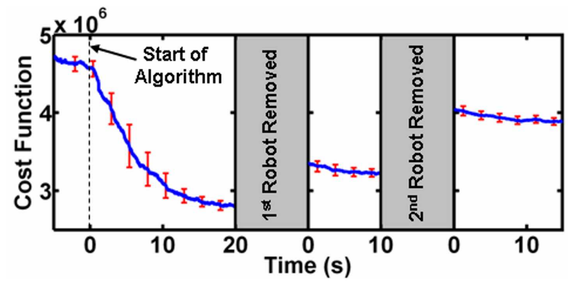


Fig. 11. The cost function during the three stages of the experiment averaged over 19 successful experiments is shown. The error bars denote one standard deviation. The experiments demonstrate the performance of the algorithm, and its ability to adapt to unforeseen robot failures.

MIT in a room equipped with a Vicon motion capture system. This system uses 16 high resolution infrared cameras to measure the global state of each robot at a rate of 120 Hz. The state update messaged are then broadcast wirelessly over 2.4 GHz Digi XBee-PRO radio modules at a rate of 50 Hz to all robots in the system, where they are parsed by the onboard microcontroller modules. In addition to using this information for the coverage controller, each module runs a PID position control loop at 33 Hz [25]. The system configuration is shown in Figure 9.

The coverage algorithm for a circular field of view using (10),(8), and (9) was implemented on each robot, running asynchronously in a fully distributed fashion. The algorithm calculated the waypoints $c(t)$ and $z(t)$ from Algorithm 1 at 1 Hz. The camera parameters were set to $a = 10e^{-6}$ and $b = 10e^{-2}$ m (which are typical for commercially available cameras), the circular field of view half angle as $\theta = 35°$, the information per area was a constant $\phi = 1$, the prior area per pixel was $w = 10e^{-6}$ m^2, and the control gain was $k = 10e{-}5$. The environment to be covered was a skewed rectangle, 3.7 m across at its widest, shown in white in Figure 10.

To test the effectiveness of the algorithm and its robustness to robot failures, we conducted experiments as follows: 1) three robots moved to their optimal positions using the algorithm, 2) one robot was manually removed from the environment, and the remaining two were left to reconfigure automatically, 3) a second robot was removed from the environment and the last one was left to reconfigure automatically.[4] Figure 10 shows photographs of a typical experiment at the beginning (Figure 10(a)), after the first stage (Figure 10(b)), after the second stage (Figure 10(c)), and after the third stage (Figure 10(d)).

[4]A video showing the indoor experiments and numerical simulations can be found at `http://people.csail.mit.edu/schwager/Movies/ICRACamerasFinal.mp4`

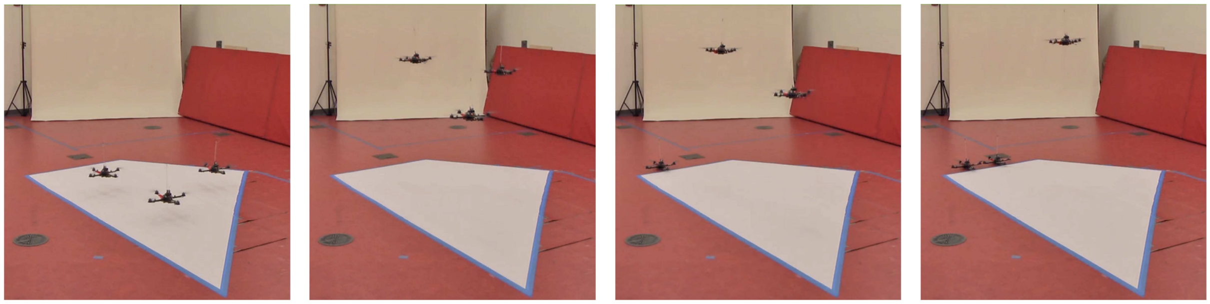
We repeated the above experiment a total of 20 times. Of these 19 runs were successful, while in one experiment two of the robots collided in midair. The collision was caused by an unreliable gyroscopic sensor, not by a malfunction of the coverage algorithm. With appropriate control gain values, collisions are avoided by the algorithm's natural tendency for neighbors to repel one another.

The coverage cost of the robots over the course of the experiment, averaged over the 19 successful experiments, is shown in Figure 11, where the error bars represent one standard deviation from the mean. Notice that when one robot is removed, the cost function momentarily increases, then decreases as the remaining robots find a new locally optimal configuration. The algorithm proved to be robust to the significant, highly nonlinear unmodeled aerodynamic effects of the robots, and to individual robot failures.

B. Optimal Coverage of an Outdoor Environment

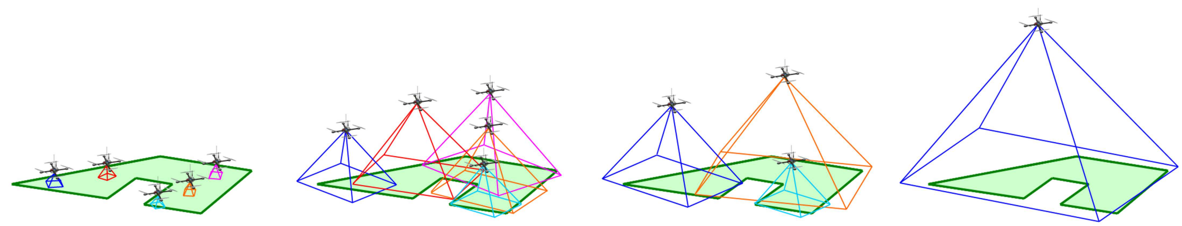
We also conducted outdoor experiments with five quadrotor robots at the German Aerospace Center, Deutsches Zentrum für Luft und Raumfahrt (DLR) in Oberpfaffenhofen, Germany. An onboard Ascending Technologies AutoPilot module stabilized each robot about a GPS and compass waypoint. In addition, state estimates were acquired from the AutoPilot module by the onboard microprocessor module at 4 Hz. Using the longer range 900 Mhz Xbee-XSC radio modules, these estimates were propagated among the group using the multi-hop algorithm in Section VI with a time slot of length $\gamma = 3$, thus forming a mobile ad hoc robot network.

The coverage algorithm for a rectangular field of view (with $\theta_1 = 35°$ and $\theta_1 = 26.25°$) using (10),(8), and (9)



(a) Initial Config. (b) Three Config. (c) Two Config. (d) One Config.

Fig. 10. Frame shots from an experiment with three AscTec Hummingbird quadrotor robots are shown. After launching from the ground (Figure 10(a)), the three robots stabilize in a locally optimal configuration (Figure 10(b)). Then one robot is manually removed to simulate a failure, and the remaining two move to compensate for the failure (Figure 10(c)). Finally a second robot is removed and the last one again moves to compensate for the failure (Figure 10(d)). The robots move so that their fields of view (which cannot be seen in the snapshots) cover the environment, represented by the white polygon.



(a) Initial Config. (b) Five Config. (c) Three Config. (d) One Config.

Fig. 12. Frame shots from an experiment with three AscTec Hummingbird quadrotor robots are shown. After launching from the ground (Figure 12(a)), the five robots stabilize in an optimal configuration (Figure 12(b)). Then two robots are manually landed to simulate failures, and the remaining three move to compensate for the missing cameras (Figure 12(c)). Finally two more robots are removed and the last one moves to view the whole environment by itself (Figure 12(d)).

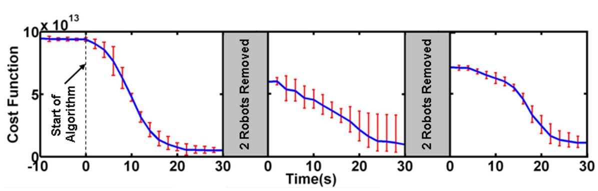
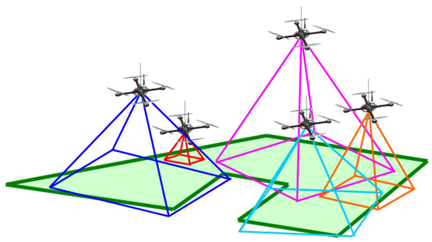


Fig. 13. The cost function during the three stages of the experiment, averaged over 10 successful experiments, is shown in Figure 13. The error bars denote one standard deviate from the mean.

was implemented on each robot running asynchronously in a fully distributed fashion. Similar to the indoor experiments, the robots were expected to cover a skewed rectangular environment measuring approximately 60 meters at its widest. In addition, a square area was removed to create a nonconvex environment. These experiments were also performed in three stages: 1) five robots moved to their optimal positions using the algorithm, 2) two robots were manually piloted away from the environment, and the remaining three were left to reconfigure automatically, 3) two more robots were manually piloted away from the environment and the last one was left to reconfigure automatically. Figure 12 shows diagrams created from acquired ground truth data of a typical experiment at the beginning (Figure 12(a)), after the first stage (Fig, 12(b)), after the second stage (Fig, 12(c)), and after the third stage (Fig, 12(d)).

The above experiment was repeated a total of 10 times, during which all robots successfully converged to their final positions for coverage. The coverage cost of the robots over the course of the experiment, averaged over the 10 experiments, is shown in Figure 13. Similarly to the indoor experiments, the mean cost decreases at each stage, then increases when robots are removed, and decreases again as the remaining robots settle into a new equilibrium. We witnessed several different equilibrium configurations for the three robot system, resulting in a large variation in local optimal cost. Several factors could have contributed to this outcome, such as GPS or compass error, network noise or latency, and variations in the initial positions of the five robots. However, for each run the system was successful in converging to an equilibrium configuration, verifying the practical



(a) Five Config.

(b) Camera Coverage

Fig. 14. This figure shows the composite view from the five cameras prior to reaching their final configuration in the first phase of the experiment. The five images overlay a larger, wider area view taken by a quadrotor robot manually flow above the system.

viability of the coverage algorithm.

To visualize the coverage, we affixed iFlip video cameras to the base of each quadrotor robot. A sixth robot was flown manually above the system to record the entire team during the experiment. Figure 14(b) shows five higher resolution views overlaying a larger aerial mosaic, with the lowest robots giving the highest resolution at ground level. Also note the difference in equilibrium configuration of Figure 14(a) when compared with Figure 12(b). This outcome was the result of a malfunctioning robot (the one indicated with the red field of view); however, its neighboring teammates shifted their position to cover where this robot normally would have gone.

VIII. Conclusions

In this paper we presented a distributed control algorithm to position robotic cameras to cover an environment. The controller is proven to locally minimize a cost function representing the aggregate information per pixel of the robots over the environment, and can be used in nonconvex and disconnected environments. We also proposed a custom networking algorithm to communicate the necessary state information among the robots. We showed simulation examples of the control algorithm running on a group of fixed cameras with rotating fixtures, a group of flying robots with downward facing cameras, and a mixed group of fixed and flying cameras all mounted on rotating fixtures. We implemented the algorithm on a group of three autonomous quadrotor robots in an indoor environment, and on a group of five autonomous quadrotor robots in an outdoor environment, and experimentally demonstrated robustness to unforeseen robot failures.

IX. Acknowledgments

This research was done in the Distributed Robotics Laboratory at CSAIL, MIT. The outdoor experiments with five robots were done at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. We would like to express our gratitude to Fabian DePonte-Müller, Martin Frassl, Michael Lichtenstern, Thomas Strang, and Kai Wendlandt of the DLR for their assistance in conducting the outdoor experiments.

X. Appendix

Proof (Theorem 1)**.** *We can break up the domain of integration into two parts as*

$$\mathcal{H} = \int_{Q\cap\mathcal{B}_i} h_{\mathcal{N}_q}\phi(q)\,dq + \int_{Q\backslash\mathcal{B}_i} h_{\mathcal{N}_q}\phi(q)\,dq.$$

Only the integrand in the first integral is a function of p_i since the condition $i \in \mathcal{N}_q$ is true if and only if $q \in \mathcal{B}_i$ (from the definition of $\mathcal{N}_q$). However the boundaries of both terms are functions of p_i, and will therefore appear in boundary terms in the derivative. Using the standard rule for differentiating an integral, with the symbol $\partial\cdot$ to mean boundary of a set, we have

$$\frac{\partial\mathcal{H}}{\partial p_i} = \int_{Q\cap\mathcal{B}_i} \frac{\partial h_{\mathcal{N}_q}}{\partial p_i}\phi(q)\,dq$$
$$+ \int_{\partial(Q\cap\mathcal{B}_i)} h_{\mathcal{N}_q}\phi(q)\frac{\partial q_{\partial(Q\cap\mathcal{B}_i)}}{\partial p_i}^T n_{\partial(Q\cap\mathcal{B}_i)}\,dq$$
$$+ \int_{\partial(Q\backslash\mathcal{B}_i)} h_{\mathcal{N}_q\backslash\{i\}}\phi(q)\frac{\partial q_{\partial(Q\backslash\mathcal{B}_i)}}{\partial p_i}^T n_{\partial(Q\backslash\mathcal{B}_i)}\,dq, \quad (35)$$

where $q_{\partial\cdot}$ is a point on the boundary of a set expressed as a function of p_i, and $n_{\partial\cdot}$ is the outward pointing normal vector of the boundary of the set. Decomposing the boundary further, we find that $\partial(Q\cap\mathcal{B}_i) = (\partial Q\cap\mathcal{B}_i)\cup(Q\cap\partial\mathcal{B}_i)$ and $\partial(Q\backslash\mathcal{B}_i) = (\partial Q\backslash\mathcal{B}_i)\cup(Q\cap\partial\mathcal{B}_i)$. But points on ∂Q do not change as a function of p_i, therefore we have

$$\frac{\partial q_{(\partial Q\cap\mathcal{B}_i)}}{\partial p_i} = 0 \quad \forall q \in \partial Q\cap\mathcal{B}_i$$
$$\text{and}\quad \frac{\partial q_{(\partial Q\backslash\mathcal{B}_i)}}{\partial p_i} = 0 \quad \forall q \in \partial Q\backslash\mathcal{B}_i.$$

Furthermore, everywhere in the set $Q \cap \partial\mathcal{B}_i$ the outward facing normal of $\partial(Q \backslash \mathcal{B}_i)$ is the negative of the outward facing normal of $\partial(Q \cap \mathcal{B}_i)$,

$$n_{\partial(Q\backslash\mathcal{B}_i)} = -n_{(\partial(Q\cap\mathcal{B}_i)} \quad \forall q \in Q \cap \partial\mathcal{B}_i.$$

Simplifying (35) leads to

$$\frac{\partial\mathcal{H}}{\partial c_i} = \int_{Q\cap\partial\mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\backslash\{i\}})\phi(q) \cdot \frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial c_i}^T n_{(Q\cap\partial\mathcal{B}_i)}\, dq. \tag{36}$$

and

$$\frac{\partial\mathcal{H}}{\partial z_i} = \int_{Q\cap\partial\mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\backslash\{i\}})\phi(q) \cdot \frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial z_i}^T n_{(Q\cap\partial\mathcal{B}_i)}\, dq - \int_{Q\cap\mathcal{B}_i} \frac{2h_{\mathcal{N}_q}^2}{a(b-z_i)^3}\phi(q)\, dq, \tag{37}$$

where we used the fact that $\partial h_{\mathcal{N}_q}/\partial c_i = [0 \quad 0]^T$, and a straightforward calculation yields $\partial h_{\mathcal{N}_q}/\partial z_i = -2h_{\mathcal{N}_q}^2/(a(b-z_i)^3)$. Now we solve for the boundary terms,

$$\frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial c_i}^T n_{(Q\cap\partial\mathcal{B}_i)} \text{ and } \frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial z_i}^T n_{(Q\cap\partial\mathcal{B}_i)},$$

which generally can be found by implicitly differentiating the constraint that describes the boundary. Henceforth we will drop the subscript on q, but it should be understood that we are referring to points, q, constrained to lie on the set $Q \cap \partial\mathcal{B}_i$. A point q on the boundary set $Q \cap \partial\mathcal{B}_i$ will satisfy

$$\|q - c_i\| = z_i \tan\theta, \tag{38}$$

and the outward facing normal on the set $Q \cap \mathcal{B}_i$ is given by

$$n_{(Q\cap\partial\mathcal{B}_i)} = \frac{(q - c_i)}{\|q - c_i\|}.$$

Differentiate (38) implicitly with respect to c_i to get

$$\Big(\frac{\partial q}{\partial c_i}^T - I_2\Big)(q - c_i) = 0,$$

where I_2 is the 2×2 identity matrix, therefore

$$\frac{\partial q}{\partial c_i}^T \frac{(q - c_i)}{\|q - c_i\|} = \frac{(q - c_i)}{\|q - c_i\|},$$

which gives the boundary terms for (36). Now differentiate (38) implicitly with respect to z_i to get

$$\frac{\partial q}{\partial z_i}^T \frac{(q - c_i)}{\|q - c_i\|} = \tan\theta,$$

which gives the boundary term for (37). The derivative of the cost function $\mathcal{H}$ with respect to p_i can now be written as in Theorem 1

Proof (Theorem 3)**.** *The proof is the same as that of Theorem 1 up to the point of evaluating the boundary terms. Equations (36) and (37) are true. Additionally the angular component is given by*

$$\frac{\partial\mathcal{H}}{\partial \psi_i} = \int_{Q\cap\partial\mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\backslash\{i\}})\phi(q) \cdot \frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial \psi_i}^T n_{(Q\cap\partial\mathcal{B}_i)}\, dq.$$

The constraint for points on the kth leg of the rectangular boundary is

$$(q - c_i)^T R(\psi_i)^T n_k = z_i \tan\theta^T n_k,$$

from (14). Differentiate this constraint implicitly with respect to c_i, z_i, and ψ_i and solve for the boundary terms to get

$$\frac{\partial q}{\partial c_i}^T R(\psi_i)^T n_k = R(\psi_i)^T n_k,$$

$$\frac{\partial q}{\partial z_i}^T R(\psi_i)^T n_k = \tan\theta^T n_k,$$

and $$\frac{\partial q}{\partial \psi_i}^T R(\psi_i)^T n_k = -(q - c_i)^T R(\psi_i + \pi/2)^T n_k,$$

where we have used the fact that

$$\frac{\partial R(\psi_i)}{\partial \psi_i} = \begin{bmatrix} -\sin\psi_i & \cos\psi_i \\ -\cos\psi_i & -\sin\psi_i \end{bmatrix} = R(\psi_i + \pi/2).$$

Break the boundary integrals into a sum of four integrals, one integral for each edge of the rectangle. The expression in Theorem 3 follows.

Proof (Theorem 4)**.** *The gradient is derived using the same method as for the previous two cases. Break the integral domain into disjoint regions as in the proof of Theorem 1 to get*

$$\frac{\partial\mathcal{H}}{\partial \rho_i} = \int_{Q\cap\partial\mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\backslash\{i\}}) \cdot \frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial \rho_i}^T n_{(Q\cap\partial\mathcal{B}_i)}\phi(q)\, dq + \int_{Q\cap\mathcal{B}_i} \frac{\partial h_{\mathcal{N}_q}}{\partial \rho_i}\phi(q)\, dq, \tag{39}$$

and

$$\frac{\partial\mathcal{H}}{\partial \psi_i^s} = \int_{Q\cap\partial\mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\backslash\{i\}}) \cdot \frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial \psi_i^s}^T n_{(Q\cap\partial\mathcal{B}_i)}\phi(q)\, dq \quad s \in \{r, p, t\}. \tag{40}$$

We will see that the three angles, ψ_i^r, ψ_i^p, and ψ_i^t, can be treated under the same form, ψ_i^s, $s = r, p, t$. Also, henceforth we drop the subscripts on q, though it should be understood that q lies in a particular set specified by the domain of integration. We now have to solve for

$$\frac{\partial h_{\mathcal{N}_q}}{\partial \rho_i}, \quad \frac{\partial q}{\partial \rho_i}^T n_{ki}, \quad \text{and} \quad \frac{\partial q}{\partial \psi_i^s}^T n_{ki} \quad k \in \{1,2,3,4\} \quad s \in \{r,p,t\}, \tag{41}$$

where n_{ki}, as was previously defined, is the outward facing normal of the boundary of the field of view over the kth leg of the trapezoid.

First, we solve for $\frac{\partial h_{\mathcal{N}_q}}{\partial \rho_i}$ using the chain rule with (5) and (27) to get

$$\frac{\partial h_{\mathcal{N}_q}}{\partial c_i} = \frac{\partial}{\partial f_i}\Big(\sum_{j=1}^{n} f_j^{-1}\Big)^{-1}\frac{\partial f(p_i,q)}{\partial c_i} = \frac{2h_{\mathcal{N}_q}^2}{a(b - \|I_{3,2}q - \rho_i\|)^3}\frac{(I_{3,2}q - \rho_i)}{\|I_{3,2}q - \rho_i\|}. \tag{42}$$

Using this in (39) gives the second term of (28) from Theorem 4.

Next we will solve for $\frac{\partial q}{\partial \rho_i}^T n_{ki}$. Notice that the normal vector n_{ki} (expressed in the global frame GF) can be obtained by expressing the pyramid face normal m_{ki} in the global frame and projecting it into the ground plane, then re-normalizing to obtain a unit vector as follows

$$n_{ki} = \frac{I_{2,3}R_i^T m_{ki}}{\|I_{2,3}R_i^T m_{ki}\|}. \tag{43}$$

Differentiate the constraint (24) with respect to ρ_i to get

$$\frac{\partial q}{\partial \rho_i}^T I_{2,3}R_i^T m_{ki} - R_i^T m_{ki} = 0. \tag{44}$$

Substitute in $I_{2,3}R_i^T m_{ki} = n_{ki}\|R_i^T m_{ki}\|$ from (43) to get

$$\frac{\partial q}{\partial \rho_i}^T n_{ki} = \frac{R_i^T m_{ki}}{\|I_{2,3}R_i^T m_{ki}\|}. \tag{45}$$

This with the expression in (40) gives (28) from Theorem 4.

Finally we solve for $\frac{\partial q}{\partial \psi_i^s}^T n_{ki}$. Differentiate the same constraint (24) with respect to ψ_i^s to get

$$m_{ki}^T R_i I_{3,2}\frac{\partial q}{\partial \psi_i^s} + m_{ki}^T \frac{\partial R_i}{\partial \psi_i^s}(I_{3,2}q - \rho_i) = 0. \tag{46}$$

now, again, substitute $I_{2,3}R_i^T m_{ki} = n_{ki}\|R_i^T m_{ki}\|$ to get

$$\frac{\partial q}{\partial \psi_i^s}^T n_{ki} = \frac{m_{ki}^T \frac{\partial R_i}{\partial \psi_i^s}(\rho_i - I_{3,2}q)}{\|I_{2,3}R_i^T m_{ki}\|}. \tag{47}$$

Using this in (40) gives the expression in (29) from Theorem 4. We solve for $\frac{\partial R_i}{\partial \psi_i^s}$ by differentiating R_i using (18) for $s = r, p, t$ to get (30) from Theorem 4.

REFERENCES

[1] O. Pizarro and H. Singh, "Toward large area mosaicing for underwater scientific applications," *IEEE Journal of Oceanic Engineering*, vol. 28, no. 4, pp. 651–672, 2003.

[2] C. Hérnandez, G. Vogiatzis, and R. Cipolla, "Multiview photometric stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 548–554, 2008.

[3] R. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456–1477, 2001.

[4] Q. Cai and J. K. Aggarwal, "Tracking human motion in structured environments using a distributed-camera system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 12, pp. 1241–1247, 1999.

[5] S. M. Khan and M. Shah, "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1355–1360, 2003.

[6] J. Black and T. Ellis, "Multi camera image tracking," *Image and Vision Computing*, no. 11, pp. 1256–1267, 2006.

[7] R. Rao, V. Kumar, and C. J. Taylor, "Planning and control of mobile robots in image space from overhead cameras," in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 18-22 2005.

[8] M. Schwager, D. Rus, and J. J. Slotine, "Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment," *International Journal of Robotics Research*, 2010, To Appear.

[9] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, April 2004.

[10] Z. Drezner, *Facility Location: A Survey of Applications and Methods*, ser. Springer Series in Operations Research. New York: Springer-Verlag, 1995.

[11] A. Ganguli, J. Cortés, and F. Bullo, "Maximizing visibility in nonconvex polygons: nonsmooth analysis and gradient algorithm design," in *Proceedings of the American Control Conference*, Portland, OR, June 2005, pp. 792–797.

[12] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *Proceedings of the IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.

[13] A. Breitenmoser, M. Schwager, J. C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *Proc. of the International Conference on Robotics and Automation (ICRA 10)*, Anchorage, Alaska, USA, May 2010, pp. 4982–4989.

[14] M. Schwager, D. Rus, and J. J. Slotine, "Decentralized, adaptive coverage control for networked robots," *International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, March 2009.

[15] S. Martínez, J. Cortés, and F. Bullo, "Motion coordination with distributed information," *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 75–88, 2007.

[16] M. L. Hernandez, T. Kirubarajan, and Y. Bar-Shalom, "Multisensor resource deployment using posterior Cramér-Rao bounds," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 2, pp. 399–416, 2004.

[17] F. Farshidi, S. Sirouspour, and T. Kirubarajan, "Optimal positioning of multiple cameras for object recognition using Cramér-Rao lower bound," in *Proc. of the IEEE International Conference on Robotics and Automation*, Orlando, Florida, 2006, pp. 934–939.

[18] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. H. ans A. Pope, R. Wildes, D. Hirvonen, M. Hansen, and P. Burt, "Aerial video surveillance and exploration," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1518–1539, 2001.

[19] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. K. Hedrick, "An overview of emerging results in cooperative UAV control,"

in *Proc. of the 43rd IEEE Conference on Decision and Control*, vol. 1, Nassau, 2004, pp. 602–607.

[20] B. Bethke, M. Valenti, and J. How, "Cooperative vision based estimation and tracking using multiple uav's," in *Advances in Cooperative Control and Optimization*, ser. Lecture Notes in Control and Information Sciences. Berlin: Springer, 2007, vol. 369, pp. 179–189.

[21] W. Ren and R. W. Beard, "Cooperative surveillance with multiple UAV's," in *Distributed Consensus in Multi-vehicle Cooperative Control*, ser. Communications and Control Engineering. London: Springer, 2008, pp. 265–277.

[22] M. Schwager, B. Julian, and D. Rus, "Optimal coverage for multiple hovering robots with downward-facing cameras," in *Proceedings of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 12–17 2009, pp. 3515–3522.

[23] B. Julian, M. Schwager, M. Angermann, and D. Rus, "A location-based algorithm for multi-hopping state estimates within a distributed robot team," in *Proceedings of the 7th International Conference on Field and Service Robotics (FSR 09)*, Cambridge, MA, July 14-16 2009.

[24] M. Schwager, "A gradient optimization approach to adaptive multi-robot control," Ph.D. dissertation, Massachusetts Institute of Technology, September 2009.

[25] B. Julian, "An embedded controller for quad-rotor flying robots running distributed algorithms," Master's thesis, Massachusetts Institute of Technology, September 2009.

[26] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, June 2008, manuscript preprint. Electronically available at http://coordinationbook.info.

[27] E. Hecht, *Optics*, 3rd ed. Reading, MA: Addison Wesley, 1998.

[28] J. LaSalle, "Some extensions of liapunov's second method," *IRE Transactions on Circuit Theory*, vol. 7, no. 4, pp. 520–527, 1960.

[29] M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*. Orlando, FL: Academic Press, Inc., 1974.

[30] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1kHz," in *Proc. of the 2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, April 2007, pp. 361–366.