2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Optimal Coverage for Multiple Hovering Robots with Downward Facing Cameras

Mac Schwager,* Brian J. Julian,*† and Daniela Rus*
*Computer Science and Artificial Intelligence Lab (CSAIL), MIT, Cambridge, MA 02139, USA
†MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA, 02420, USA
Email: schwager@mit.edu, bjulian@mit.edu, and rus@csail.mit.edu

***Abstract*— This paper presents a distributed control strategy for deploying hovering robots with multiple downward facing cameras to collectively monitor an environment. Information per pixel is proposed as an optimization criterion for multi-camera placement problems. This metric is used to derive a specific cost function for multiple downward facing cameras mounted on hovering robot platforms. The cost function leads to a gradient-based distributed controller for positioning the robots. A convergence proof using LaSalle's invariance principle is given to show that the robots converge to locally optimal positions. The controller is demonstrated in experiments with three flying quad-rotor robots.**

I. Introduction

Multiple collaborating robots with cameras are useful in a broad range of applications, from surveying disaster sites, to observing the health of coral reefs. However, an immediate and difficult question arises in such applications: how should one position the robots so as to maintain the best view of an environment? In this paper we offer an approach 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 develop the approach in detail for the particular case of multiple downward facing cameras mounted to robots. The cost function leads to a gradient-based distributed controller for the robots to position themselves in three dimensions so as to best observe a planar environment over which they hover. We present simulation results in a Matlab environment. We also present experimental results with three AscTec Hummingbird quad-rotor robots. This paper is accompanied by a video of the quad-rotors using the control algorithm.

Our algorithm can be used in support of a higher-level computer vision task, such as object recognition or tracking. We address the problem of how to best position the robots given that the data from their cameras will be used by some computer vision algorithm. Our design principle can be readily adapted to a number of applications. For example, it could be used to control groups of autonomous underwater or air vehicles to do mosaicing [1], or to produce photometric stereo from multiple camera views [2], for inspection of underwater or land-based archaeological sites, biological environments such as coral reefs or forests, disaster sites, 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 aid in navigation of agents on the ground [7].

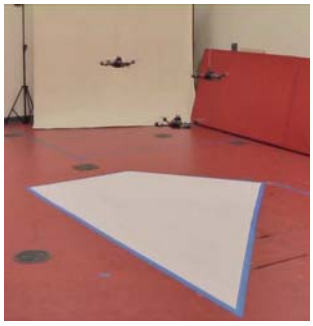


Fig. 1. This snapshot of an experiment shows three flying quad-rotor robots moving so that their cameras cover the environment represented by the white polygon.

The main contributions of this work are three-fold: 1) we propose the minimum information per pixel principle as a cost function for camera placement and formulate it explicitly for the case of multiple hovering robots with downward facing cameras, 2) we use the cost function to design a controller to deploy multiple robots to their optimal positions in a distributed fashion, 3) we implement the proposed controller on three quad-rotor robots. The proposed robot coordination algorithm is fully decentralized, provably stable, adaptive to a changing number of flying agents and a changing environment, and will work with a broad class of environment geometries, including convex, non-convex, and disconnected spaces.

We are inspired by a recent body of work concerning the optimal deployment of robots for providing sensor coverage of an environment. Cortés et al. [8] introduced a stable distributed controller for sensor coverage based on ideas from

The authors would like to thank an anonymous reviewer who gave particularly helpful and detailed comments.

This work was supported in part by the MURI SWARMS project grant number W911NF-05-1-0219, NSF grant numbers IIS-0513755, IIS-0426838, CNS-0520305, CNS-0707601, EFRI-0735953, MIT Lincoln Laboratory, the MAST project, and the Boeing Corporation.

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.

the optimal facility placement literature [9]. This approach involves a Voronoi partition of the environment, and has seen several extensions ([10]–[12]). One recent extension described in [13], 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. 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 constraint that the environment must be convex (to prevent agents from trying to leave the environment). In contrast, we consider agents moving in a space $\mathbb{R}^3$, covering an arbitrary lower dimensional environment $Q \subset \mathbb{R}^2$. This eliminates the need for 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 [14], and an application of the idea for cameras was given in [15]. We choose to focus on the problem of positioning downward facing cameras, similarly to [16], as opposed to arbitrarily oriented cameras. Many geometrical aspects of the problem are significantly simplified in this setting, yet there are a number of practical applications that stand to benefit from controlling cameras in this way, as previously described. 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 [17], or [18] and [19] for two recent examples.

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 image that is exposed to a given pixel. 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 [11]). 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 [9], [20]. 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 $\mathbb{R}^3$, its orientation in $so(3)$ (the three rotational angles), and perhaps a lens zooming parameter in the interval $(0, \infty)$, 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 consider the special case in which the camera is downward facing (hovering over Q). Indeed, this case is of particular interest in many applications, as described in Section I. 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 Fig. 2. In Section III-A we will consider a camera with a rectangular field of view, but initially consider a circular field of view, so the rotational orientation of the downward facing camera is irrelevant. 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 center point of the field of view, and $z \in \mathbb{R}$ the height 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.

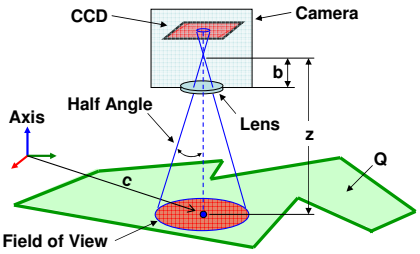


Fig. 2. The camera optics and the geometry of the environment are shown in this figure.

To find the area per pixel function, $f(p, q)$, consider the geometry in Fig. 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 to be $b^2/(b-z)^2$) times the area of one pixel [21]. 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 (we will avoid dealing with infinite quantities in the multi-camera case). 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 Fig. 3. Intuitively, an

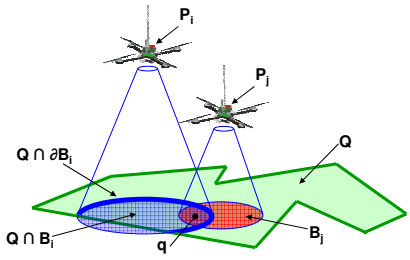


Fig. 3. This figure shows the relevant quantities involved in characterizing the intersecting fields of view of two cameras.

area of Q that is being observed by two 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 (assuming the cameras are identical, so they use the same function $f(p_i, q)$) 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 sensor fusion.

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, $\phi(q)$, which could also be derived from a pre-existing photograph. 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

$$h_{\mathcal{N}_q}(p_1, \ldots, p_n, q) = \Big(\sum_{i \in \mathcal{N}_q} f(p_i, q)^{-1} + w^{-1}\Big)^{-1}. \tag{4}$$

to give the cost function

$$\mathcal{H}(p_1, \ldots, p_n) = \int_Q h_{\mathcal{N}_q}(p_1, \ldots, p_n, q)\phi(q)\, dq. \tag{5}$$

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

$$\min_{(p_1,\ldots,p_n)\in\mathcal{P}^n} \mathcal{H}. \tag{6}$$

The cost function (5) 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 can can swivel on gimbels). We proceed with the special case of downward facing cameras, where $\mathcal{P} = \mathbb{R}^3$ and $f(p_i, q)$ is from (3) for the remainder of the paper.

III. Distributed Control

We will take the gradient of (5) and find that it is distributed among agents. This will lead to a 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

$$\frac{\partial \mathcal{H}}{\partial c_i} = \int_{Q \cap \partial \mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q \setminus \{i\}}) \frac{(q - c_i)}{\|q - c_i\|} \phi(q)\, dq, \tag{7}$$

and

$$\frac{\partial \mathcal{H}}{\partial z_i} = \int_{Q \cap \partial \mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q \setminus \{i\}}) \phi(q) \tan\theta\, dq - \int_{Q \cap \mathcal{B}_i} \frac{2h_{\mathcal{N}_q}^2}{a(b - z_i)^3} \phi(q)\, dq. \tag{8}$$

Please refer to the appendix for a proof.

Remark 1 (Intuition): We will consider a controller that moves a robot in the opposite direction of its gradient component. In which case, the single integral for the lateral component (7) 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 (8) 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 (7) and vertical (8) components can be computed by robot i with knowledge of 1) its own position, p_i, 2) the extent of 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 $\mathcal{N}_i = \{j \mid Q \cap \mathcal{B}_i \cap \mathcal{B}_j \neq \emptyset, i \neq j\}$. The controller can be run over a network that is a subgraph of the required proximity graph, in which case performance will degrade gracefully as the network becomes more sparse.

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

$$u_i = -k \partial \mathcal{H} / \partial p_i, \tag{9}$$

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,

$$\dot{p}_i = u_i, \tag{10}$$

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

Theorem 2 (Convergence): For a network of n robots with the dynamics in (10), using the controller in (9),

$$\lim_{t \to \infty} \frac{\partial \mathcal{H}}{\partial p_i} = 0 \quad \forall i \in \{1, \ldots, n\}. \tag{11}$$

Proof: The proof is an application of LaSalle's invariance principle ([22], [20] Theorem 1.17 [1]). 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 continuously differentiable function of p_j for all j (i.e. all terms in the Hessian of $\mathcal{H}$ are continuous). Taking the time derivative of $\mathcal{H}$ along the trajectories of the system gives

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

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. ■

To be more precise, there may exist configurations at which $\frac{\partial \mathcal{H}}{\partial p_i} = 0 \ \forall i$ that are saddle points or local maxima of $\mathcal{H}$. However, since the controller is a gradient controller, only the local *minima* of $\mathcal{H}$ are stable equilibria. A proof of this intuitively obvious fact about gradient systems can be found in [23], Chapter 9, Section 4.

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 (9) is then given in Algorithm 1.

Remark 4 (Time Complexity): 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 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 $T(n, m) \in O(n^2 m)$.

Remark 5 (Adaptivity): The controller is adaptive in the sense that it will stably reconfigure if any number of robots

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

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

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

Compute and move to

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

end loop

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 proof, but has been shown to be the case in simulation studies.

Remark 6 (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.

A. Rectangular Field of View

Until this point we have assumed that the camera's field of view, $\mathcal{B}_i$ is a circle, which eliminates a rotational degree of freedom. Of course, actual cameras have a rectangular CCD array, and therefore a rectangular field of view. In this section we revisit the gradient component in Theorem 1 and calculate it for a rectangular field of view and a robot with a rotational degree of freedom.

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 rotation angle. Define a rotation matrix

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

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

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

where $\theta = [\theta_1, \theta_2]^T$ is a vector with two angles which are the two half-view angles associated with two perpendicular edges of the rectangle, as shown in Fig. 4, and the $\le$ symbol applies element-wise (all elements in the vector must satisfy $\le$). 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 $n_4 = [0 \quad -1]$. The cost function, $\mathcal{H}(p_1, \ldots, p_n)$, is the

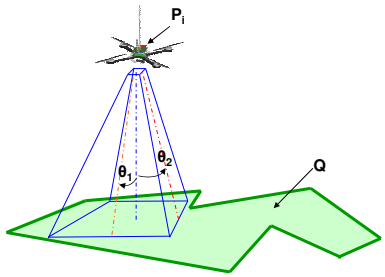


Fig. 4. The geometry of a camera with a rectangular field of view is shown in this figure.

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

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

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

and

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

Remark 7 (Intuition): 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

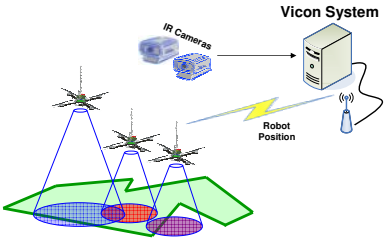


Fig. 5. This figure shows the experimental setup. The robots positions were captured with an Vicon motion capture system. The robots used their position information to run the coverage algorithm in a distributed fashion.

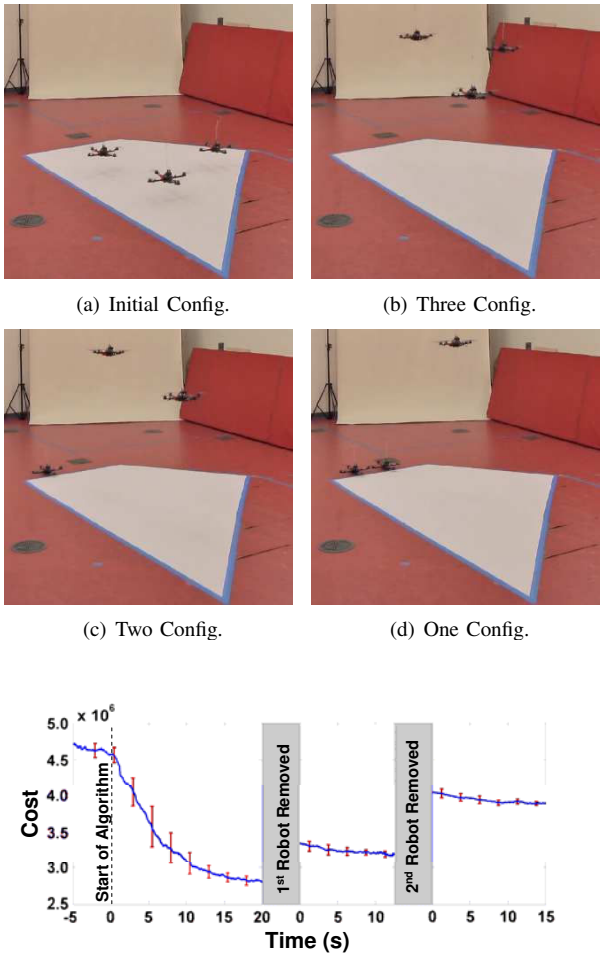
field of view into the environment, while also rotating away from other robots whose field of view intersects its own. Computation of the gradient component for the rectangular field of view is of the same complexity as the circular case, and carries the same constraint on the communication topology.

IV. Experiments

We implemented Algorithm 1 on a group of three AscTec Hummingbird flying quad-rotor robots. Our experiments were performed at CSAIL, MIT in a laboratory equipped with a Vicon motion capture system. The robots' position coordinates (x, y, z, yaw) were broadcast wirelessly at 50Hz via a 2.4 Ghz xBee module. Each robot was equipped with a custom ARM microprocessor module running a PID position control loop at 33Hz. Pitch and roll were fully stabilized by the commercial controller described in [24]. A schematic of the experimental setup is shown in Fig. 5.

The coverage algorithm was implemented on the same onboard ARM modules, running asynchronously in a fully distributed fashion. The algorithm calculated way points ($c_i(t)$ and $z_i(t)$ from Algorithm 1) at 1Hz. This time-scale separation between the coverage algorithm and the PID controller was required to approximate the integrator dynamics assumed in (10). The camera parameters were set to $a = 10^{-6}$ and $b = 10^{-2}$m (which are typical for commercially available cameras), the field of view was $\theta = 35$deg, the information per area was a constant $\phi(q) = 1$, the prior area per pixel was $w = 10^{-6}\text{m}^2$, and the control gain was $k = 10^{-5}$. The enviornment to be covered was a skewed rectangle, 3.7m across at its widest, shown in white in Fig. 6.

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. Fig. 6 shows photographs of a typical experiment at the beginning (Fig. 6(a)), after the first stage (Fig. 6(b)), after the second stage (Fig. 6(c)), and after the third stage (Fig. 6(d)). The coverage cost of the robots over the course of the whole experiment, averaged over 19 experiments,

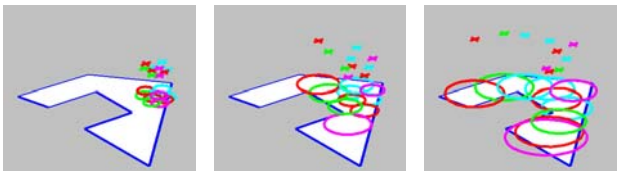


(a) Initial Config. (b) Three Config. (c) Two Config. (d) One Config. (e) Cost Function

Fig. 6. Frame shots from an experiment with three AscTec Hummingbird quad-rotor robots are shown. After launching from the ground (Fig. 6(a)), the three robots stabilize in an optimal configuration (Fig. 6(b)). Then one robot is manually removed to simulate a failure, and the remaining two move to a new optimal position (Fig. 6(c)). Finally a second robot is removed and the last one stabilizes at an optimal position (Fig. 6(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. The cost function during the three stages of the experiment, averaged over 19 successful experiments, is shown in Fig. 6(e). The error bars denote one standard deviation. The experiments demonstrate the performance of the algorithm, and its ability to adapt to unforeseen robot failures.

is shown in Fig. 6(e), where the error bars represent one standard deviation. Notice that when one robot is removed, the cost function momentarily increases, then decrease as the remaining robots find a new optimal configuration. The algorithm proved to be robust to the significant, highly nonlinear unmodeled aerodynamic effects of the robots, and to individual robot failures. This paper is accompanied by a video showing the experiments and numerical simulations.

We repeated the above experiment a total of 20 times. Of these 19 were successful, while in one experiment two of the robots collided in mid air. 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



(a) Initial Config. (b) Middle Config. (c) Final Config.

Fig. 7. Results of a simulation with ten robots covering a nonconvex environment are shown. The $\times$'s mark the robot positions and the circles represent the fields of view of their cameras. Communication failures and noise on the robots' velocities are also modeled in the simulation. The initial, middle, and final configurations are shown in 7(a), 7(b), and 7(c), respectively.

neighbors to repel one another.

V. Simulations

We conducted numerical simulations to investigate the scalability and robustness of the algorithm. Hovering robots with integrator dynamics (10) were simulated using Algorithm 1 on a centralized processor. The values of a, b, θ, ϕ, w, and k were the same as in the experiments. The simulations were over a non-convex environment, as shown in Fig. 7. Communication constraints were modeled probabilistically. The probability of robot i communicating with robot j was calculated as a linear function of the distance between them decreasing from 1 at a distance of 0, to 0 at a distance of $R = 1.5m$, and the environment width was roughly $3m$. Uncertainty in the robots' velocity was modeled as white Gaussian noise with covariance of $I_3 \times 10^{-4} m^2/s^2$ (where I_3 is the 3×3 identity matrix). Fig. 7 shows the results of a typical simulation with ten robots. The robots start in an arbitrary configuration and spread out and up so that their fields of view cover the environment.

VI. Conclusion

In this paper we presented a distributed control algorithm to allow hovering robots with downward facing 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 implemented the algorithm on a group of three autonomous quad-rotor robots, and experimentally demonstrated robustness to unforseen robot failures. We also investigated scalability and robustness to network failures in simulations with ten flying robots. This paper is accompanied by a video showing robot experiments and simulations.

VII. Appendix

Proof: [Theorem 1] We can break up the domain of integration into two parts as

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

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

$$\begin{aligned}\frac{\partial\mathcal{H}}{\partial p_i} &= \int_{Q\cap\mathcal{B}_i} \frac{\partial h_{\mathcal{N}_q}}{\partial p_i}\phi(q)\,dq \\ &+ \int_{\partial(Q\cap\mathcal{B}_i)} h_{\mathcal{N}_q}\phi(q)\frac{\partial q_{\partial(Q\cap\mathcal{B}_i)}}{\partial p_i}^T n_{\partial(Q\cap\mathcal{B}_i)}\,dq \\ &+ \int_{\partial(Q\setminus\mathcal{B}_i)} h_{\mathcal{N}_q\setminus\{i\}}\phi(q)\frac{\partial q_{\partial(Q\setminus\mathcal{B}_i)}}{\partial p_i}^T n_{\partial(Q\setminus\mathcal{B}_i)}\,dq, \end{aligned} \quad (18)$$

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 $\partial(Q\setminus\mathcal{B}_i) = (\partial Q\setminus\mathcal{B}_i)\cup(Q\cap\partial\mathcal{B}_i)$. But points on ∂Q do not change as a function of p_i, therefore we have

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

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

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

Simplifying (18) leads to

$$\begin{aligned}\frac{\partial\mathcal{H}}{\partial c_i} &= \int_{Q\cap\partial\mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\setminus\{i\}})\phi(q) \\ &\cdot\frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial c_i}^T n_{(Q\cap\partial\mathcal{B}_i)}\,dq. \end{aligned} \quad (19)$$

and

$$\begin{aligned}\frac{\partial\mathcal{H}}{\partial z_i} &= \int_{Q\cap\partial\mathcal{B}_i} (h_{\mathcal{N}_q} - h_{\mathcal{N}_q\setminus\{i\}})\phi(q) \\ \cdot\frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial z_i}^T n_{(Q\cap\partial\mathcal{B}_i)}\,dq &- \int_{Q\cap\mathcal{B}_i} \frac{2h_{\mathcal{N}_q}^2}{a(b-z_i)^3}\phi(q)\,dq, \end{aligned} \quad (20)$$

where we used the fact that $\partial h_{\mathcal{N}_q}/\partial c_i = [0 \;\; 0]^T$, 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

$$\|q - c_i\| = z_i\tan\theta, \quad (21)$$

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 (21) 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 (19). Now differentiate (21) 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 (20). 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 (19) and (20) are true. Additionally the angular component is given by

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

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. ■

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] 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.

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

[10] 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.

[11] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

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

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

[24] 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.