Multi-robot Task Assignment for Aerial Tracking with Viewpoint Constraints

Ray, Aaron ; Pierson, Alyssa; Zhu, Hai; Alonso-Mora, Javier; Rus, Daniela

DOI
10.1109/IROS51168.2021.9636719

Publication date
2021

Document Version
Final published version

Published in
Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2021)

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636719

Multi-robot Task Assignment for Aerial Tracking with Viewpoint Constraints

Aaron Ray[1], Alyssa Pierson[1], Hai Zhu[2], Javier Alonso-Mora[2], Daniela Rus[1]

***Abstract*— We address the problem of assigning a team of drones to autonomously capture a set desired shots of a dynamic target in the presence of obstacles. We present a two-stage planning pipeline that generates offline an assignment of drone to shots and locally optimizes online the viewpoint. Given desired shot parameters, the high-level planner uses a visibility heuristic to predict good times for capturing each shot and uses an Integer Linear Program to compute drone assignments. An online Model Predictive Control algorithm uses the assignments as reference to capture the shots. The algorithm is validated in hardware with a pair of drones and a remote controlled car.**

I. INTRODUCTION

We wish to develop algorithms for coordinating heterogeneous systems of aerial and ground agents when the ground agents are not cooperating with the aerial vehicles. One class of problems within this broad scope is following a ground-based moving agent (e.g. robot or human) with an aerial vehicle, subject to constraints such as "keep a certain feature of the agent in the field of view while avoiding environmental obstacles". This problem is challenging because it requires a real time adaptive solution for the local control with global objectives and constraints. Practical and robust solutions will enable new applications such as autonomous drone videography that go beyond today's recording capabilities. Current drone videography can follow an actor. In this paper we describe a solution that supports finer-grain specifications, such as "keep the actor's face in the field of view". The solution has to combine real-time local response with global planning to accommodate the presence of obstacles (e.g. avoid bridges).

More specifically, we enable a team of videography drones to autonomously track and capture a sequence of desired shots of a moving target such as a ground robot or a person. Framing a scene for videography is a complicated process that depends on a range of aesthetic preferences of the videographer. However, many of the important framing primitives can be distilled into a small set of parameters that define the camera's desired viewpoint, such as size of the target in the frame, position of the target in the image, and position of the camera relative to the target. We would like for a videographer to be able to specify a set of desired shots based on these parameters, and have the team of drones determine the best trajectories for capturing the video of the subject as it moves through the environment. We assume access to a good prediction of the subject's motion, as otherwise there is little use in pre-planning long sequences of shots.

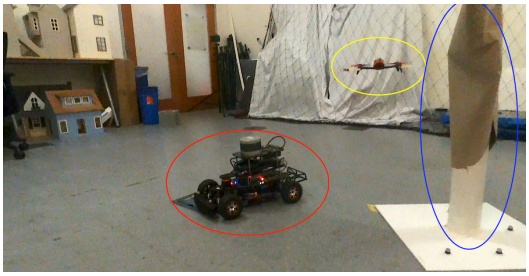


Fig. 1. Viewpoint from a videography drone looking at a target (red). Another drone (yellow) optimizes the capture of a different shot while avoiding occlusions from obstacles (blue).

We present a two-stage viewpoint optimization pipeline that enables a team of drones to capture a series of shots that match desired aesthetic qualities. A high-level planner uses a visibility heuristic and an Integer Linear Program (ILP) optimization routine to choose when each drone should capture which shot. This assignment results in a reference trajectory that can be tracked by an online Model Predictive Control (MPC) algorithm with a cost function based on the specified viewpoint parameters. The controller can locally optimize the drones' trajectories to account for stochastic target motion. We demonstrate a videography scenario with a pair of drones assigned to capture several shots of a remote controlled racecar in the presence of ellipsoidal obstacles.

Related work A large body of literature exists related to defining shot aesthetics for a videography drone [1]. These works explore parameterizing desired shot qualities and controls [2][3][4][5], feasibility of dynamic shots [6], and assigning sequences of shots [7][8]. Other work as focused on algorithmic frameworks for helping directors achieve desirable aesthetic qualities of their shots[9][10].

Drone videography requires precise motion planning and control algorithms to achieve the defined aesthetic objectives. One common approach solves a constrained nonlinear optimization in a receding horizon fashion [11], [12], [13]. Other approaches optimize for trajectory smoothness [14], focus on a series of static landmarks [15], or use deep reinforcement learning [16]. More generally, the problem of tracking multiple subjects is similar to persistent monitoring [17], [18]; patrolling and surveillance [19], [20], [21], [22]; and pursuer-evader games [23], [24], [25] .

Contributions We build upon the authors' previous work in [12] and focus on the problem of optimizing sequences of shots from multiple cameras and perspectives subject to constraints. In contrast to [12] and other recent work [16][26]

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA {aray, apierson, rus}@csail.mit.edu

[2]Cognitive Robotics, Delft University of Technology, 2628 CD, Delft, Netherlands {H.Zhu, J.AlonsoMora}@tudelft.nl

This work supported in part by the Office of Naval Research (ONR), and Singapore's DSO National Laboratory. Their support is gratefully acknowledged.

that has focused on reactively finding good viewpoints in unstructured scenes, we assume that the operational environment is known and the subject's motion can be estimated ahead of time. In this context, a director may have a set of desired shots that should be captured during the scene. It is nontrivial to decide when each shot should be taken so that the shots are minimally obstructed and the ordering is feasible for the drones. A higher-level global planner is used to augment the online local planner and ensure the desired viewpoints can all be captured. The main contributions of the paper are:

- Presenting a novel high-level videography planner based on a visibility heuristic to globally optimize ordering of shots of a dynamic target
- Showing that the reference trajectory generated by this assignment can be followed by a viewpoint-aware receding horizon controller to locally optimize shot aesthetics
- Demonstrating these algorithms in hardware experiments with a pair of drones and remote control racecar videography target

The remainder of this paper is organized as follows: we briefly summarize shot aesthetic specifications and formalize the minimization of the viewpoint costs in Section II. Section III presents the high-level global planner which generates a sequence of shots and reference trajectory to guide each drone. In Section IV we explain the videography MPC formulation used for local viewpoint optimization. Section V describes our experimental results.

II. Problem Definition

When a predictable subject is operating in a known environment, we would like to be able to coordinate a set of desired shots such that each shot is captured with minimal occlusion. For example, repeated takes on a movie set or a well-trodden mountain bike path are scenarios where the subject's motion is fairly restricted and can be estimated ahead of time, even if the exact path is not known. Existing videography approaches focus on reactive planning to maintain good views of a target. We are more interested in global planning that ensures a team of videography drones sequences the set of desired shots to ensure they are all fit in within the allotted time and with minimal obstruction.

A. Preliminaries

Throughout this paper vectors are denoted in bold lowercase letters, $\mathbf{w}$, matrices in plain uppercase, M, and sets in calligraphic, $\mathcal{S}$. $\|\mathbf{w}\|$ denotes the Euclidean norm of $\mathbf{x}$ and $\|\mathbf{w}\|_Q^2 = \mathbf{w}^T Q \mathbf{w}$ denotes the weighted squared norm. We use matrices $Q_{\_}$ that appear in these weighted norms are used to represent tuneable parameters in cost functions that allow the user to express the relative importance of different desiderata.

We assume that n_d videography drones operate in an environment with known set of obstacles $\mathcal{O}$. While the MPC formulation we use requires obstacles to be represented as ellipsoids, our main contribution of the higher-level planner supports any obstacle set representation that allows for computing ray intersections efficiently.

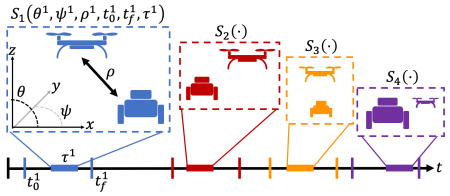


Fig. 2. Illustration of the shot schematic. Within each shot, we optimize based on viewpoint costs. We also optimize the shot assignment for the sequence of shots between drones.

We also assume access to a probability distribution over the videography subject's future trajectory, denoted $\mathbb{Y}$. We sample from this distribution to predict good times for shots to be taken.

B. Defining Desired Shots

Much of the defining aesthetic of a camera framing is determined by:

1) **Orientation** of the camera relative to the film subject, denoted by azimuth and elevation, (ψ, θ),
2) **Subject's Distance** to the camera, ρ,
3) **Subject's Position** in the frame, (C_x, C_y).

as presented in [12]. We also note that it is usually desirable for the videography subject not to be occluded in the image.

In addition to the spatial viewpoint parameters, we consider a time window within which a shot should be taken, $[t_0, t_f]$, and a desired duration τ^i.

Definition 1 (Shot Specification)**.** *A shot specification* $S_i = (\theta^i, \psi^i, \rho^i, C_x^i, C_y^i, t_0^i, t_f^i, \tau^i)$ *is set of viewpoint parameters and timing constraints define each shot. We denote the set of desired shot specifications* $\mathcal{S}$.

Figure 2 illustrates these parameters. The viewpoint properties depend on the drone state $\mathbf{x}_b$, target state $\mathbf{x}_t$, and obstacles in the environment $\mathcal{O}$. We briefly define a cost function associated with each parameter, and refer to [12] for derivation. Let $\mathbf{r}_{ct}$ and $\mathbf{r}_{ct}^c$ denote the vector from the camera origin to the target expressed in the world and camera reference frames respectively. Let $\mathbf{r}_d^c$ denote the desired vector from the camera origin to target as defined by image position parameters C_x and C_y. Let $\boldsymbol{\alpha}_d$ represent the unit vector corresponding to the desired angles (θ, ψ) for the drone relative to the target. With these definitions, we can define cost functions that penalize a drone's state as it deviates from the desired viewpoint. Let c_{image} denote the cost for the target's position in the image:

$$c_{image} = \left\| \frac{\mathbf{r}_d^c}{\|\mathbf{r}_d^c\|} - \hat{\mathbf{r}}_{ct}^c \right\|_{Q_i}.$$

The cost c_{scale} penalizes distance deviation from the target,

$$c_{scale} = \left\| \|\mathbf{r}_{ct}\| - \rho \right\|_{Q_s},$$

and c_{angle} is a cost that depends on the deviation of the drone's angles relative to the target:

$$c_{angle} = \left\| -\frac{\mathbf{r}_{ct}}{\|\mathbf{r}_{ct}\|} - \frac{\alpha_d}{\|\alpha_d\|} \right\|_{Q_a}.$$

Finally, $c_{occlusion}$ penalizes drone positions that have an obstructed view of the target. This cost is based on

the position of the target compared to the occlusion cone associated with the camera and each obstacle. We define a cost based on d_v, the distance by which the target is within an obstacle's occlusion cone. Let $\mathbf{r}_{co}^o$ represent the vector from the camera to an obstacle in the obstacle's reference frame. We define $c_{occlusion} = \max(0, d_v)^2$ where

$$d_v = \frac{\mathbf{r}_{co}^{oT}\mathbf{r}_{ct}^o}{\sqrt{\|\mathbf{r}_{co}^o\|^4 - \|\mathbf{r}_{co}^o\|^2}} - \sqrt{\|\mathbf{r}_{ct}^o\|^2 - \frac{(\mathbf{r}_{co}^{oT}\mathbf{r}_{ct}^o)^2}{\|\mathbf{r}_{ct}^o\|^2}},$$

if $\mathbf{r}_{ct}^{oT}\mathbf{r}_{co}^o / \|\mathbf{r}_{co}^o\| > \|\mathbf{r}_{co}^o\| - 1$. Otherwise, the target is closer to the camera than the obstacle is and $c_{occlusion} = 0$. Each obstacle contributes a separate occlusion cost. A similar visibility cone concept can be used to penalize mutual visibility between drones as in [13], although we do not employ that cost here.

C. Defining Desired Sequences

These costs c aid in quantifying the quality of a single shot. Our viewpoint objective is to minimize these costs over a sequence of shots.

Definition 2 (Shot Sequence)**.** *A shot sequence $A_i = (S_j^{t_a}, S_k^{t_b}, \ldots)$ is the set of all shots assigned to drone i. Each assigned shot $S_j^{t_a}$ corresponds to a shot specification S_j and a time when the drone should be capturing that shot, t_a. A_i^j denotes the j-th shot assigned to drone i. The set of all assigned shots across all drones and times is $\mathcal{A} = \cup_i A_i$.*

For each shot, we define a vector of cost functions $\mathbf{J}_i^j$ that penalizes drone i's state from deviating from the parameters defined by shot S_j. $\mathbf{J}_i^j$ contains penalties for deviating from the desired location of the target in the image plane, its size in the image plane, and the camera position relative to the target. $\mathbf{J}_i^j$ also penalizes drone states that have an obstructed view of the target. Intuitively,

$$\mathbf{J}_i^j = [c^i_{\text{image}}, c^i_{\text{scale}}, c^i_{\text{angle}}, c^i_{\text{occlusion}}]^T. \tag{1}$$

Given the trajectory of the videography target, $\mathbf{J}_i^j$ is a function of drone i's state and time. For a given set of shots $\mathcal{S}$, our goal is to then find the trajectories for all drones such that $\mathbf{J}$ is minimized over all shots.

Problem 1. *For a desired set of shots $\mathcal{S}$ and a target trajectory Y, we define the total videography cost L as a function of the trajectories of the n_d drones:*

$$L(\mathbf{x}_1, \ldots, \mathbf{x}_{n_d}) = \sum_{j=0}^{|S|} \min_{t_0^j \le t \le t_f^j} \min_i \int_t^{t+\tau^j} \left\| \mathbf{J}_i^j(\mathbf{x}_i(t), t) \right\|_{Q_x} dt. \tag{2}$$

In general, Y is stochastic, and by extension $\mathbf{J}_i^j$ is as well. We seek a control policy for each drone that will minimize the expected videography cost L.

We use a two stage approach to find good policies for minimizing L. First, a high-level planning algorithm uses a simple heuristic to predict good times for capturing each shot. A discrete optimization algorithm assigns each drone to a sequence of shots that minimizes the heuristic cost. Next, the optimized assignment and expected target trajectory are used to generate a reference trajectory for a videographic Model Predictive Control (MPC) algorithm that locally optimizes the aesthetic parameters online.

III. SHOT ASSIGNMENT

The high-level shot planning is carried out by a centralized algorithm that finds a sequence of shots for each drone to capture, as well as a reference trajectory to follow. The algorithm requires the desired set of shots $\mathcal{S}$, a distribution over target trajectories $\mathbb{Y}$, and a set of obstacles in the environment $\mathcal{O}$. It relies on a heuristic to choose a low-cost reference trajectory that guarantees visibility of the target. The heuristic provides an estimate of the best times to start capturing each shot. We sample the lowest-cost times for each shot and assign each drone a sequence of shots and times based on an ILP minimization.

A. Assignment Heuristic

We seek a simple heuristic for each shot i that maps time to a reference position, while ensuring that the target is visible. We refer to the position given by this heuristic as $\hat{\mathbf{x}}^i(t)$, and $H^i(t)$ as the associated cost. By focusing on a single cost and position at each time, the problem of optimizing shot cost over all possible trajectories simplifies to choosing a sequence of shots and times for each drone.

The target is visible from the drone if and only if a ray cast from the target to the drone hits no obstacles before reaching the drone. This observation inspires a heuristic for the reference shot position — cast a ray from the target's position $\mathbf{x}_t$ in the desired shot direction, stopping at the desired shot distance or the intersection with an obstacle, whichever is first. Let $\mathbf{x}^*(t) = \mathbf{x}_t(t) + \rho^i \boldsymbol{\alpha}$ and $\hat{\mathbf{x}}(t) = \mathbf{x}_t(t) + d\boldsymbol{\alpha}$ where

$$\boldsymbol{\alpha} = [\cos\theta\cos(\psi + \psi_t), \cos\theta\sin(\psi + \psi_t), \sin\theta]^T,$$

and d is the smaller of ρ_i and the closest obstacle intersection along $\boldsymbol{\alpha}$. Intuitively, $\mathbf{x}^*$ is the drone position that would have zero viewpoint cost, and $\hat{\mathbf{x}}$ is as close as we can get to that position along the desired direction from the target, before hitting an obstacle. We similarly define $\tilde{\mathbf{x}}(t)$ as a ray cast from $\mathbb{E}(\mathbf{x}(t))$ in the expected direction of $\boldsymbol{\alpha}$. The path $\tilde{\mathbf{x}}$ will be used as the reference path to transition between shots. As the reference position is constructed such that it satisfies the desired relative angles ψ and θ, c_{scale} is the only nonzero viewpoint cost. The reference trajectory is also guaranteed to have visibility of the target. For shot i, we consider a heuristic videography cost H^i_{vid} that is a function of only time:

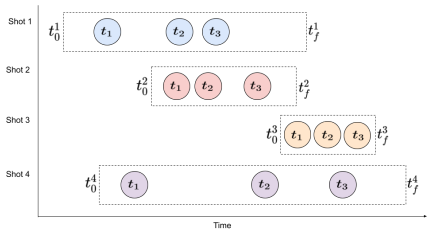
$$H^i_{vid}(t) = \|\mathbf{x}^*(t) - \hat{\mathbf{x}}(t)\|.$$

The position heuristic $\hat{\mathbf{x}}$ does not enforce continuity of the trajectory, so we add an additional discontinuity cost H_{dis} for each shot:

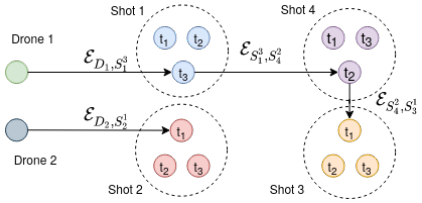
$$H^i_{dis} = q_{dis} \left\| \frac{d\hat{\mathbf{x}}}{dt} \right\|^2,$$

where q_{dis} is a tuning parameter that adjusts the importance of ensuring the reference trajectory can be perfectly tracked.

Together, H_{vid} and H_{dis} sum to the heuristic for instantaneous cost, H_{fine}. Each shot S_i lasts for a duration τ^i. We define a shot cost H^i_{shot} that estimates the cost incurred by



(a) An example of shots four shots, S_1, S_2, S_3, S_4. Three times with low heuristic cost H_{shot} have been sampled for each shot, as discussed in Section III-A. In reality, many more times are sampled. Our experiments use 20 samples per shot.



(b) An example of a solution to the PDAG-Minimum-Paths problem for $m = 2$, where the edges not on a solution path have been omitted. Note that each color set is on exactly one path, and there are two paths. This diagram also exemplifies how a minimization in the form of (3) maps to an instance of PDAG-Minimum-Paths. The edge labels $\mathcal{E}$ demonstrate the semantics of the construction of $\mathcal{G}_{shot}$. Note that the three times chosen for Drone 1's assignment are in ascending order in (a).

Fig. 3. An illustrative example of how (a) a set of desired shots are considered at discrete times and (b) turned into a graphical optimization problem.

capturing shot S_i starting at time t by setting $H^i_{shot}(t)$ to be the average value of H_{fine} in the interval $[t, t + \tau_i]$. We take the expectation over the target's trajectory distribution and calculate it by sampling.

We can now use this heuristic H_{shot} to find good times to capture each shot by finding an assignment of shots to drones such that the sum of H_{shot} over all of the chosen shot times is minimized and each drone can be expected to feasibly capture all of its assigned shots. The size of the optimization problem depends on how finely we sample H_{shot}. We reduce the number of times considered for starting each shot by sampling p random time indices $\mathbf{t}^i_{sampled}$ for each shot, biasing the samples toward lower-cost times. The sampling process greatly reduces the size of the problem.

B. Choosing the Best Shots

An assignment of drones to shots can now be computed by finding an assignment that minimizes the total heuristic cost incurred. Recall from Definition 2 that $S^{t_a}_j$ denotes the shot j that has been assigned to begin at time t_a, A_i denotes the sequence of shots that have been assigned to drone i, and $\mathcal{A} = \cup_i A_i$ is the set of all assigned shots and times. We must search for an assignment $\mathcal{A}$ that minimizes the total sum of shot costs, while ensuring that each drone has enough time to transition between the shots it has been assigned.

The distance between shots is not deterministic, as it depends on the target's stochastic trajectory. Moreover, even if the drone cannot fully complete the transition between shots in the allotted time, its actual position at the beginning of the next shot may be as good as the desired reference position $\hat{\mathbf{x}}(t)$, especially if it can get close to $\hat{\mathbf{x}}$. Instead of hard transition feasibility constraints, we introduce a cost function that penalizes transitions that may be dynamically infeasible. For a pair of shot assignments $S^{t_a}_j, S^{t_b}_k$, the transition cost H_{trans} is defined as

$$H_{trans}(S^{t_a}_j, S^{t_b}_k) = \mathbb{E}(\left\|[\max(0, d_{trans} - d_{max}), d_{trans}]^T\right\|_{Q_t}),$$

where d_{max} is the maximum distance the drone can travel between time t_a and t_b and d_{trans} is the distance required to transition between the end of shot $S^{t_a}_j$ and the beginning of shot $S^{t_b}_k$. The weighting Q_t is set such that H_{trans} incurs a very large penalty when the transition distance between shots is expected to be too long (i.e. $d_{trans} > d_{max}$), and a smaller penalty based on d_{trans} which encourages assignments that result in the drones traveling shorter distances.

In addition to this cost term, we require that if one shot precedes another in the assignment ordering, the first shot must end before the second begins. We define slightly different notation for the heuristic cost H_{shot} to refer to the cost of an assignment rather than the cost of a shot. For assignment $A^j_i = S^{t_a}_k$, we notate the assignment cost as $H(A^j_i) = H^k(t_a)$. We want to solve the following minimization:

$$\mathcal{A}^* = \underset{\mathcal{A}}{\operatorname{argmin}} \sum_{i=1}^{n_d} \left(\sum_{j=1}^{|A_i|} H_{shot}(A^j_i) + \sum_{j=2}^{|A_i|} H_{trans}(A^{j-1}_i, A^j_i) \right) \tag{3a}$$

$$\text{subject to} \quad \forall s \in S, \quad s \in \mathcal{A}, \tag{3b}$$

$$|S| = |\mathcal{A}|, \tag{3c}$$

$$\forall S^t_j \in A, \quad t \in \mathbf{t}^j_{sampled}, \tag{3d}$$

$$\forall (S^{t_a}_j, S^{t_b}_k) \in A_i, \quad t_b \geq t_a + \tau^j. \tag{3e}$$

Constraints (3b) and (3c) ensure that each shot appears exactly once in the ordering. Constraints (3d) and (3e) force the chosen times to be from $\mathbf{t}_{sampled}$ and satisfy the temporal consistency previously discussed.

The cost associated with (3) acts like an upper bound on (2). For each shot, the assignment associates a time window, a drone, and a heuristic reference path that minimizes most of the constituent costs in $\mathbf{J}$ (the exception being c_{scale}). The cost is not truly an upper bound because $\hat{\mathbf{x}}$ may not result in a feasible trajectory, but it provides intuition for why we expect minimizing the cost in (3) will provide a good reference trajectory for minimizing the final videography cost from (2).

C. Constructing a Graph Formulation

To find a minimizing solution to (3), we draw inspiration from minimum-cost flow assignment methods [27]. While this assignment problem cannot actually be solved with minimum-cost flow solvers, encoding it in a graph yields a straightforward interpretation as a constrained minimum-cost path problem in a Partitioned Directed Acyclic Graph (PDAG), which can be solved with an ILP.

Definition 3 (PDAG)**.** *A Partitioned Directed Acyclic Graph (PDAG) is a Directed Acyclic Graph $\mathcal{G}$, such that each vertex is assigned one of k labels.*

For this purposes of illustration, we will refer to these labels of the PDAG by k different colors. We now define Problem 2 on finding a set of minimum-cost paths through the PDAG.

Problem 2 (PDAG-Minimum-Paths)**.** *Consider a weighted PDAG, $\mathcal{G}$. Given a maximum number of paths m, PDAG-Minimum-Paths(m, $\mathcal{G}$) is the problem of finding the lowest-weight set of paths $\mathcal{P}$ such that $|\mathcal{P}| \leq m$ and exactly one vertex of each color is on some path or determining that there is no such satisfying $\mathcal{P}$.*

Figure 3 demonstrates how a solution to a PDAG-Minimum-Paths problem with $m = 2$ encodes the shot assignment problem. By design, a path must visit each of the graph partitions, which means that it passes through every color label of the graph. We now discuss how to construct a PDAG that encodes the constraints of (3).

Note that (3e) implies that the assignment of shots must satisfy a strict partial ordering that ensures the sequence is monotonically increasing in time and has no shot overlaps assigned to a single drone. We use this partial order to generate a DAG $\mathcal{G}_{shot}$, where vertices represent each S_i^t, for $t \in \mathbf{t}^i_{sampled}$. This selection of vertices enforces (3d). We choose the color label of each vertex $S_i^t \in \mathcal{G}_{shot}$ to be its shot index i. We also add a vertex in $\mathcal{G}_{shot}$ for each of n_d drones that can be assigned a shot. These drone vertices are connected to all existing vertices in $\mathcal{G}_{shot}$ and each have a unique color. The directed edge between $S_j^{t_a}$ and $S_k^{t_b}$ (if it exists) is denoted $\mathcal{E}_{S_j^{t_a} S_k^{t_b}}$ with cost given by

$$H(\mathcal{E}_{S_j^{t_a} S_k^{t_b}}) = H_{shot}(S_k^{t_b}) + H_{trans}(S_j^{t_a}, S_k^{t_b}). \quad (4)$$

The cost for edges connected to the drone nodes is denoted

$$H(\mathcal{E}_{D_i, S_k^{t_b}}) = H_{shot}(S_k^{t_b}). \quad (5)$$

We now present Proposition 1, which connects $\mathcal{G}_{shot}$ to (3).

Proposition 1.
Solutions to PDAG-Minimum-Paths(n_d, G_{shot}) also minimize (3)*.*

Proof. We will show that the constraints and costs in (3) are equivalent to the constraints and costs of solutions to PDAG-Minimum-Paths(n_d, G_{shot}) (PMP). Let each shot $s \in \mathcal{S}$ correspond to a color in G_{shot}. If we consider the set of assigned shots $\mathcal{A}$ as corresponding to the set of vertices in G_{shot} that are on paths selected by PMP, then enforcing each shot $s \in S$ to appear in the assignment A as in (3b) is equivalent to enforcing that every color appears on the solution to PMP. Enforcing $|\mathcal{S}| = |\mathcal{A}|$ as in (3c) is then equivalent to restricting each color to appear at most once on the PMP solution. The vertices of G_{shot} are chosen by construction to respect constraint (3d). The connectivity of G_{shot} was defined by the partial ordering implied by (3e), so every solution to PMP respect this ordering constraint. Each path in the solution of PMP corresponds to the assignment ordering A_i for a drone. The cost of the path consists of the shot cost connected to each vertex and the transition cost connected to each edge, as in (4) and (5), the same expressions used to express the costs of assignments A_i in (3a). As the constraints and costs are equivalent, the solution to PDAG-Minimum-Paths(n_d, G_{shot}) also minimizes (3). □

D. Solving PDAG-Minimum-Paths

We now formulate PDAG-Minimum-Paths as an ILP. Consider a weighted PDAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. To solve PDAG-Minimum-Paths($n_d, \mathcal{G}$), we first construct a modified graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ which is the same as $\mathcal{G}$, but it contains an extra n_d vertices. Each of the extra vertices is connected to all of the original vertices, and they each have a unique color. These additional vertices are denoted $\hat{\mathcal{V}}$.[1]

We construct an ILP that contains a variable x_i for each edge $e_i \in \mathcal{E}'$. The cost of each variable C_x is the same as the cost of its corresponding edge. Let $\hat{\mathcal{X}}$ denote the set of variables corresponding to edges connected to $\hat{\mathcal{V}}$. Let $\mathcal{X}_{*,i}$ denote the set of variables corresponding to edges incident on V_i. Let $\mathcal{X}_{i,*}$ denote the set of variables corresponding to edges directing away from V_i. Let $\mathcal{X}^k$ denote the set of variables corresponding to edges incident on color k. The ILP solving PDAG-Minimum-Paths($D, \mathcal{G}$) can be defined as:

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \sum_{x \in \mathcal{X}} x C_x \quad (6a)$$

$$\text{s.t. } \forall x \in \mathcal{X}, x \in \{0, 1\}, \quad (6b)$$

$$\forall k, \sum\nolimits_{i=1}^{|\mathcal{X}^k|} \sum\nolimits_{x \in \mathcal{X}^k_{*,i}} x = 1, \quad (6c)$$

$$\forall i, \sum\nolimits_{x \in \mathcal{X}_{i,*}} x \leq \sum\nolimits_{x \in \mathcal{X}_{*,i}} x \quad (6d)$$

$$\forall i, \sum\nolimits_{x \in \hat{\mathcal{X}}_{i,*}} x \leq 1. \quad (6e)$$

The edges corresponding to variables with a value of one in $\mathcal{X}^* \setminus \hat{\mathcal{X}}$ are considered to be the set of edges in the solution to PDAG-Minimum-Paths. If there is no satisfying assignment for $\mathcal{X}^*$, then there is no solution to PDAG-Minimum-Paths.

Proposition 2. *The Integer Linear Program in* (6) *generates a solution PDAG-Minimum-Paths, which minimizes* (3)*.*

Proof. Constraint (6c) ensures that the total number of edges incident on a color set is equal to one. This satisfies that all colors are part of some path. Constraints (6d) and (6e) restrict the solution set to contain at most D paths, by limiting the number of selected edges attached to $\hat{\mathcal{V}}$ to D and requiring that all other edges start at a node with an incident edge. Thus, every solution of (6) encodes a valid set of paths in PDAG-Minimum-Paths. By the construction of (6), every PDAG-Minimum-Paths problem can be represented by the ILP. Thus, (6) solves PDAG-Minimum-Paths, and by Proposition 1, it minimizes (3) as well. □

The size of the ILP necessary to solve the drone assignment depends on the number of shots $|\mathcal{S}|$, the number of samples we choose per shot p, and the number of drones, n_d. The ILP has variables corresponding to transitions between the $O(|\mathcal{S}|p)$ sampled times of the set of shots. For each drone, there is also a variable associated with each sampled time. This results in a ILP with $O((|\mathcal{S}|p)^2 + |\mathcal{S}|pn_d)$

[1]Note that in the case of the drone assignment problem, the "drone" vertices that were added can be considered as $\hat{\mathcal{V}}$.

variables. There area total of $2|\mathcal{S}|p+|\mathcal{S}|$ constraints. We find in practice that the number of shots we can define is limited by the horizon over which we can make predictions of the target's motion rather than by computational burden of the assignment algorithm.

The full high-level shot assignment pipeline is summarized in Algorithm 1.

Algorithm 1 Videography Assignment

procedure ASSIGNSHOTS($\mathcal{S}, \mathbb{Y}, \mathcal{O}$)
  for $i = 1 : |\mathcal{S}|$ **do**
    $H^i_{\text{shot}} \leftarrow$ CalculateShotHeuristic($\mathbb{Y}, \mathcal{O}, S_i$)
    $\mathbf{t}^i_{\text{sampled}} \leftarrow$ ImportanceSample(H^i_{shot})
  for each $(S^{t_a}_i, S^{t_b}_j) \in$ PotentialShotTransitions **do**
    $H_{\text{trans}}(S^{t_a}_i, S^{t_b}_j) \leftarrow \mathbb{E}(\text{TransitCost}(S^{t_a}_i, S^{t_b}_k, \mathbb{Y}))$
  $\mathcal{A}^* \leftarrow$ IntegerProgram($t_{\text{sample}}, H_{\text{sample}}, H_{\text{trans}}, \mathcal{S}$)
  return $\mathcal{A}^*$

IV. ONLINE VIEWPOINT OPTIMIZATION

The assignment $\mathcal{A}^*$ from (3) is used to guide the online MPC. For each shot assignment $S^{t_a}_j \in \mathcal{A}^*_i$, drone i will attempt to minimize the viewpoint cost for shot j in the range $[t_a, t_a + \tau^j]$. In the intermediate time between successive assignments $S^{t_a}_j$ and $S^{t_b}_k$, the MPC follows a reference path that leads it from the expected end of one shot to the expected beginning of another, $\tilde{\mathbf{x}}(t_a + \tau^j)$ to $\tilde{\mathbf{x}}(t_b)$. We precompute a Probabilistic Roadmap (PRM)[28] of free space in the environment, and use it to compute the reference path between $\tilde{\mathbf{x}}(t_a+\tau^j)$ and $\tilde{\mathbf{x}}(t_b)$. During each shot period, the MPC locally optimizes the viewpoint costs. The MPC solves for drone trajectories and gimbal controls in real-time separately for each drone. The rest of the MPC formulation is materially the same as the method presented in [12], but we summarize it here for completeness. For speed considerations and analysis of the nonlinear videography MPC performance, we refer to [12].

A. Drone and Camera Model

We assume the drone's motion model follows some nonlinear differential equation $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, with $\mathbf{x}(0) = \mathbf{x}_0$ and $\mathbf{x}^k \in \mathcal{X} \subset \mathbb{R}^{n_x}$ denotes the state of the drone and $\mathbf{u}^k \in \mathcal{U} \subset \mathbb{R}^{n_u}$ the control inputs at time step k. $\mathcal{X}$ and $\mathcal{U}$ are the admissible state space and control space, respectively. While $\mathbf{x}$ has been used in previous sections to denote only position, here it represents the full state. $\mathbf{x}_0$ is the initial state of the drone. Our experiments use the Parrot Bebop 2 quadrotor with its integrated front camera and electronic gimbal. It accepts desired roll and pitch angles, yaw rate, vertical velocity, and gimbal angles as input.

B. MPC Formulation

For each drone in the team, we formulate a receding horizon constrained optimization problem with N time steps and planning horizon $N\Delta t$, where Δt is the sampling time. The optimization seeks to minimize the weighted squared norm of a cost vector $\left\|\hat{\mathbf{J}}\right\|^2_Q$. Recall that $\mathbf{J}^j_i$ denotes the instantaneous viewpoint cost incurred by drone i capturing shot j. As desired viewpoint and drone are unambiguous in the MPC formulation, we drop the sub- and superscripts in this section. We augment $\mathbf{J}$ to include the control inputs, distance from reference state $\mathbf{x}_{ref}$, and collision avoidance slack variables $\mathbf{S}$:

$$\hat{\mathbf{J}}^k = [\mathbf{J}^T, \mathbf{u}^T, \|\mathbf{x}_{ref} - \mathbf{x}\|, \mathbf{S}^T]^T|_{t=k}.$$

A matrix of weightings Q defines the relative importance of each cost function. When the drone is assigned to take a shot, the weighting matrix does not penalize reference position error. When the drone transitions between shots, the relative viewpoint and distance are not penalized, although the image position cost c_{image} is kept. This keeps the drones pointing toward the target while repositioning for the next shot.

We use a third order collocation method that implicitly represents the drone's trajectory as a third order spline and the control inputs as piecewise linear functions of time as described in [29]. The dynamics constraints are applied at the collocation points $t_{c,n}$, and the drone state. The full MPC optimization can be written as:

$$\min_{\mathbf{x}^{1:N}, \mathbf{u}^{1:N}} \quad \sum_{k=1}^{N-1} \left\|\hat{\mathbf{J}}^k\right\|^2_Q + \left\|\hat{\mathbf{J}}^N\right\|^2_{Q_f} \tag{7a}$$

$$\text{s.t.} \quad \mathbf{x}^1 = \mathbf{x}(0), \tag{7b}$$

$$\dot{\mathbf{x}}(t_{c,n}) = f(\mathbf{x}(t_{c,n}), \mathbf{u}(t_{c,n})), \tag{7c}$$

$$\mathbf{x}^k \in \mathcal{X}, \tag{7d}$$

$$\mathbf{u} \in \mathcal{U}, \tag{7e}$$

$$\forall k \in \{1, \ldots, N\}. \tag{7f}$$

At each time step, the drone solves the formulated nonlinear constrained optimization problem online to generate a local trajectory and executes the first time step controls in a receding horizon fashion. The optimization is performed using CasADi [30] for automatic differentiation and IPOPT [31] for optimization. Note that the optimization is not necessarily solved to completion — the current iterate after 40 ms of computation is used to select the next control output.

In order to estimate the costs at each timestep in the horizon, the drone must have an estimate for the future trajectories of dynamic obstacles and the videography target. Our prediction model assumes constant linear and angular velocity in the target's body frame, based on current estimated velocity. For our experiments the velocity estimate is provided by a motion capture system.

We implement the MPC for multiple drones in an asynchronous manner: the MPC is solved independently for each drone, and the expected trajectory each drone generates is used to compute collision avoidance by future iterations of the other drones' MPC as described in detail in [32].

V. EXPERIMENTAL RESULTS

We validated the tracking system with a pair of Bebop 2 drones tracking a human-driven RC racecar[2]. In order to generate the predictive target distribution necessary to calculate expected shot costs, the racecar was driven around two obstacles in a loosely defined pattern for ten warmup

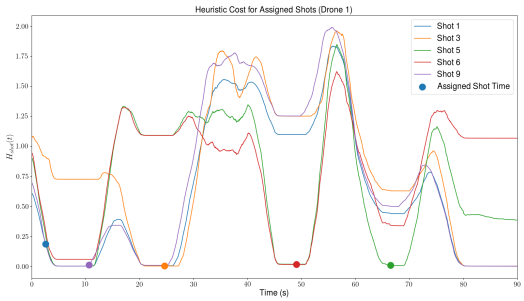
[2] https://mit-racecar.github.io/

laps. Figure 6 shows the paths from these ten runs and their mean. Note that the car's speed varies within each run and comes to a stop at several locations. The warmup runs were used to model a multivariate Gaussian distribution over the car's trajectory. Nine desired shots were specified for the drone, varying in length from four to six seconds and all constrained to occur within the ninety second duration of the car's lap. The image position parameters C_x and C_y were set the center the target in the frame in all shots. The settings for the other parameters are shown in Figure 5. The high-level shot planner assigned each drone a sequence of shots. The cost $H_{shot}(t)$ for heuristic position $\hat{\mathbf{x}}(t)$ for each shot is shown in Figure 4. The planner is able to find an assignment of shots to the drones such that the expected viewpoint cost for each shot is quite low. We note that the planner assigns overlapping shots to the two drones resulting in a sequence of shots that would have been infeasible with a single drone.

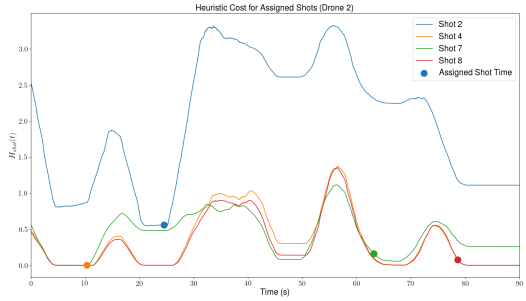
The resulting assignments were tracked over a series of ten trials in which the racecar was manually driven along the same route as the warmup laps. Figure 6 shows the trajectories for the drones during one of the runs, in addition to the car's nominal path and two obstacles. The figure also demonstrates that the planning algorithm's penalty for transition distance between shots H_{trans} leads to the assigned shots being spatially segmented between the drones, reducing the total distance each has to travel.

Figure 5 illustrates the relative viewpoint achieved by each drone relative to the desired parameters during each shot. During each assigned shot window (denoted in green), the viewpoint parameters (in blue) are close to their desired values (in orange). The high-level planning heuristic and reference trajectory places the drones near the desired viewpoints, so in many cases the desired viewpoint is reached before the shot even starts. The drones are able to achieve lower-deviation trajectories during some shots compared to others. The higher-deviation shots occur when the desired drone position is obstructed (because of a static obstacle or other drone), or if the trajectory necessary to follow the desired position is dynamically infeasible (most commonly when the car is turning quickly). Even in cases where there is significant deviation from the desired viewpoint, such as the distance in Drone 1's third shot (at about T = 25s in (5(e))), we note that the other two parameters have low deviation and the drone maintains an unoccluded view of the target as shown in the accompanying video. Instances where the observed viewpoint deviations are greater than expected from the heuristic costs in Figure 4 are mostly due to the drones acting more conservatively when close to obstacles than expected by the planner.

The performance bottleneck for the high-level assignment algorithm is solving the ILP. The size of the ILP depends on the number of shots and the number of sampled times per shot. Our scenario contained nine shots, with twenty times sampled for each shot. The resulting ILP solved in under thirty seconds on a laptop with a 2.6 GHz Intel Core i7-9750H 6-Core processor and 16 GB RAM. Different balances between optimality and solution speed can be achieved by tuning the number of sampled times. We note that while the ILP does not run in real time, it would have had time to run at least once during the scenario to reassign shots based

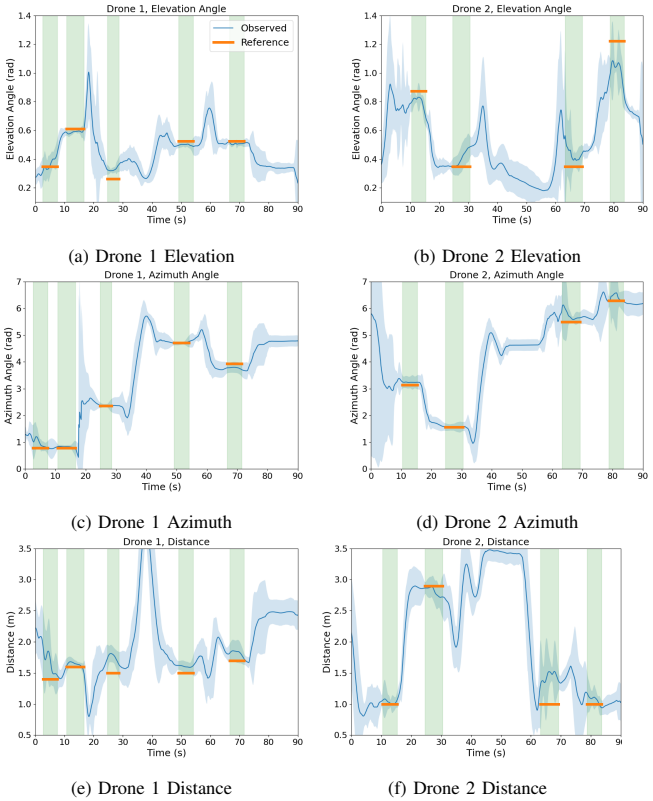


(a) Expected cost $H_{shot}(t)$ of path heuristic $\hat{\mathbf{x}}(t)$ for each of the five shots ultimately assigned to Drone 1.



(b) Expected cost $H_{shot}(t)$ of path heuristic $\hat{\mathbf{x}}(t)$ for each of the four shots ultimately assigned to Drone 2.

Fig. 4. Cost associated with heuristic path over time for each desired shot. The solid circles denote the time that the high level planner determined was the lowest expected cost.



(a) Drone 1 Elevation

(b) Drone 2 Elevation

(c) Drone 1 Azimuth

(d) Drone 2 Azimuth

(e) Drone 1 Distance

(f) Drone 2 Distance

Fig. 5. Performance of the two drones. Active shots are shaded in green, and orange bars represent the reference. The solid blue line represents the mean value over the 10 trials, and the transparent blue denotes one standard deviation of the data from the 10 trials. The drones converge to the reference values while in a shot.

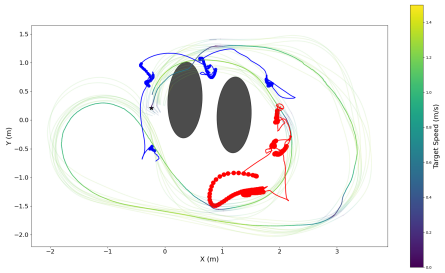


Fig. 6. The predicted mean of the racecar trajectory (opaque blue/yellow curve) and its training runs (transparent blue/yellow curves). Car speed shown on color axis. The drone trajectories for one run are shown in red and blue. Circular markers denote times when the drones were taking a shot.

on updated predictions of the target trajectory.

VI. CONCLUSIONS

This paper has demonstrated a practical algorithm for assigning a team of drones to capture a series of desired shots and locally optimizing the drone trajectories to ensure each shot is captured as well as possible. Our experiments have shown that the planning and control pipeline works on physical systems in the presence of obstacles and uncertainty over the videography target's trajectory. Future work will explore an increased uncertainty of the target's trajectory, as well as multi-target tracking.

REFERENCES

[1] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous UAV Cinematography: A Tutorial and a Formalized Shot-Type Taxonomy," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–33, Sept. 2019. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3362097.3347713

[2] Q. Galvane, C. Lino, M. Christie, J. Fleureau, F. Servant, F.-l. Tariolle, and P. Guillotel, "Directing Cinematographic Drones," *ACM Transactions on Graphics*, vol. 37, no. 3, pp. 1–18, July 2018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3243123.3181975

[3] M. Christie, P. Olivier, and J.-M. Normand, "Camera control in computer graphics," *Computer Graphics Forum*, vol. 27, no. 8, pp. 2197–2218, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2008.01181.x

[4] M. Gleicher and A. Witkin, "Through-the-lens camera control," *SIGGRAPH Comput. Graph.*, vol. 26, no. 2, p. 331–340, July 1992. [Online]. Available: https://doi.org/10.1145/142920.134088

[5] C. Lino and M. Christie, "Intuitive and efficient camera control with the toric space," *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 82:1–82:12, July 2015. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2809654.2766965

[6] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "Shot Type Feasibility in Autonomous UAV Cinematography," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 1937–1941, iSSN: 1520-6149.

[7] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, "High-Level Multiple-UAV Cinematography Tools for Covering Outdoor Events," *IEEE Transactions on Broadcasting*, vol. 65, no. 3, pp. 627–635, Sept. 2019.

[8] S. M. Drucker and D. Zeltzer, "Intelligent camera control in a virtual environment," in *In Proceedings of Graphics Interface '94*, 1994, pp. 190–199.

[9] C. Lino, M. Christie, R. Ranon, and W. Bares, "The Director's Lens: An Intelligent Assistant for Virtual Cinematography," in *ACM Multimedia*, Scottsdale, United States, Nov. 2011. [Online]. Available: https://hal.inria.fr/hal-00646398

[10] H. Jiang, B. Wang, X. Wang, M. Christie, and B. Chen, "Example-driven virtual cinematography by learning camera behaviors," *ACM Trans. Graph.*, vol. 39, no. 4, July 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392427

[11] B. Sabetghadam, A. Alcántara, J. Capitán, R. Cunha, A. Ollero, and A. Pascoal, "Optimal Trajectory Planning for Autonomous Drone Cinematography," in *2019 European Conference on Mobile Robots (ECMR)*, Sept. 2019, pp. 1–7.

[12] T. Nageli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-Time Motion Planning for Aerial Videography With Dynamic Obstacle Avoidance and Viewpoint Optimization," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, July 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7847361/

[13] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–10, July 2017. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3072959.3073712

[14] C. Gebhardt, S. Stevšić, and O. Hilliges, "Optimizing for aesthetically pleasing qadrotor camera motion," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–11, July 2018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3197517.3201390

[15] K. Xie, H. Yang, S. Huang, D. Lischinski, M. Christie, K. Xu, M. Gong, D. Cohen-Or, and H. Huang, "Creating and chaining camera moves for quadrotor videography," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–13, July 2018.

[16] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, "Autonomous aerial cinematography in unstructured environments with learned artistic decision-making," *Journal of Field Robotics*, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21931

[17] S. Alamdari, E. Fata, and S. L. Smith, "Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 138–154, 2014. [Online]. Available: https://doi.org/10.1177/0278364913504011

[18] P. Tokekar and V. Kumar, "Visibility-based persistent monitoring with robot teams," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 3387–3394.

[19] C. Pippin, H. Christensen, and L. Weiss, "Performance based task assignment in multi-robot patrolling," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 70–76. [Online]. Available: https://doi.org/10.1145/2480362.2480378

[20] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[21] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: Review," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 187–198, 2018.

[22] R. S. de Moraes and E. P. de Freitas, "Multi-uav based crowd monitoring system," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 2, pp. 1332–1345, 2020.

[23] M. A. Vieira, R. Govindan, and G. S. Sukhatme, "Scalable and practical pursuit-evasion with networked robots," *Intelligent Service Robotics*, vol. 2, no. 4, p. 247, 2009.

[24] A. Pierson, Z. Wang, and M. Schwager, "Intercepting rogue robots: An algorithm for capturing multiple evaders with multiple pursuers," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 530–537, 2017.

[25] Z. Zhou, J. R. Shewchuk, D. Stipanović, H. Huang, and C. J. Tomlin, "Smarter lions: Efficient cooperative pursuit in general bounded arenas," *SIAM Journal on Control and Optimization*, vol. 58, no. 2, pp. 1229–1256, 2020.

[26] A. Bucker, R. Bonatti, and S. Scherer, "Do you see what i see? coordinating multiple aerial cameras for robot cinematography," 11 2020, preprint on webpage at https://arxiv.org/abs/2011.05437.

[27] J. B. O. Ravindra K. Ahuja, Thomas L. Magnanti, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Inc, 1993.

[28] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[29] R. Tedrake, "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832)," chapter 10. [Online]. Available: http://underactuated.mit.edu/

[30] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[31] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[32] H. Zhu and J. Alonso-mora, "Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, apr 2019.