Vision-Based Control for Fast 3-D Reconstruction With an Aerial Robot

Eric Cristofalo, *Student Member, IEEE*, Eduardo Montijano, *Member, IEEE*, and Mac Schwager, *Member, IEEE*

***Abstract*— We propose an active perception controller to drive an aerial robot to localize 3-D features in an environment using an onboard monocular camera. The robot estimates feature positions with either an extended Kalman filter (EKF) or an unscented Kalman filter (UKF). For each filter, we derive a controller that seeks the most valuable robot motions for estimating the 3-D positions of features in the local robot body frame. The control algorithm uses an explicit expression for the gradient of the error covariance matrices for both the EKF and UKF at the next time step. This gradient approach is demonstrated in both simulations and real-world experiments on a quadrotor with a downward facing camera where it is shown to outperform a wide variety of the state-of-the-art active sensing strategies. Our proposed control algorithms are computationally efficient, making them well-suited for fast, online reconstructions onboard microaerial vehicles.**

***Index Terms*— Robot vision systems, unmanned autonomous vehicles, visual servoing.**

I. Introduction

IN THIS paper, we consider the question, what is the best instantaneous control for a microaerial vehicle (MAV) with a camera to quickly produce the most accurate 3-D reconstruction of the unknown scene? We propose an active estimation solution to find the 3-D position of features in the scene by repeatedly moving the MAV toward the negative gradient of the features' estimation error. We make no assumptions on the structure of the scene beforehand and only require that feature points can be extracted and tracked from live images. We design this active sensing control strategy using two common nonlinear filter architectures: the extended Kalman filter (EKF) and the unscented Kalman filter (UKF). The 3-D features are always measured and maintained with respect to the current camera frame. Our intention is not to build a global map for high-fidelity reconstruction, but rather to quickly obtain a coarse, local map for fast decision making on-the-fly. For example, we may want to quickly label objects for further investigation, eliminating those that do not appear to be of interest [1]. We alternatively may want to determine if an object is a friend or a foe, in order to decide an appropriate action to take next. Finally, we may want to rapidly inspect the extent and geometry of an obstacle, so that we can evasively maneuver around it. Throughout the work, our emphasis is on fast, on-board computation of control actions that provide the best improvement on the construction of the 3-D point clouds.

We derive expressions for the gradient of the error covariance matrix with respect to the camera position for the EKF and UKF, which can be used for any general nonlinear active estimation problems. We also provide the specific form of the gradients for the special case of the 3-D vision application. We then demonstrate the entire reconstruction pipeline both in simulations with quadrotor dynamics and in hardware experiments with a quadrotor flying in a laboratory environment. Our results indicate that the proposed active estimation significantly outperforms several state-of-the-art strategies. Specifically, our contributions in this paper are as follows.

1) We derive the analytic form of the gradient of the error covariance matrices for both the EKF and the UKF for use in general nonlinear active sensing problems, which, to the best of our knowledge, has not been explicitly presented in the literature for both filters.
2) We apply this active sensing algorithm to estimate the 3-D positions of features from a monocular camera on-board a quadrotor MAV.
3) We provide quantitative analysis through simulations and experiments that show our active estimation method outperforms four competing methods, including the state-of-the-art receding horizon and active structure from the motion methods.

The organization in the rest of this paper is as follows. We formalize the general active sensing problem in Section II. We review the EKF and UKF filters, and give the analytical form of the gradient of their error covariance matrices in Section III. We pose the particular problem of vision-based control for the active estimation of 3-D feature positions with an MAV in Section IV. Finally, Section V presents the results

Manuscript received February 16, 2018; revised November 11, 2018; accepted February 11, 2019. Date of publication April 2, 2019; date of current version June 11, 2020. Manuscript received in final form March 11, 2019. This work was supported in part by the Spanish Projects DPI2015-69376-R (MINECO/FEDER) and DGA T04-FSE, in part by the NSF under Grant CNS-1330008 and Grant IIS-1646921, in part by ONR under Grant N00014-18-1-2830 and Grant N62909-19-1-2027, and in part by the Ford-Stanford Alliance Program. The work of E. Cristofalo was supported by the National Defense Science and Engineering Graduate (NDSEG) Fellowship. Recommended by Associate Editor B. Jayawardhana. *(Corresponding author: Eric Cristofalo.)*

E. Cristofalo and M. Schwager are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA (e-mail: ecristof@stanford.edu; schwager@stanford.edu).

E. Montijano is with the Departamento de Informática e Ingeniería de Sistemas, Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, 50009 Zaragoza, Spain (e-mail: emonti@unizar.es).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2019.2905227

of simulations and experiments with a quadrotor, and our conclusions are summarized in Section VI.

A. Related Work

Obtaining the unknown depth of image features has been a recurring problem in both the computer vision and robotics communities. Structure from motion (SfM) [2] and simultaneous localization and mapping (SLAM) [3] are two widely used solutions to this problem, for which numerous variations have been proposed. For example, in the former, there are solutions that minimize the error between reprojected image patches to estimate camera motion [4], or represent image pixel depth as a probabilistic map [5]. Examples of the latter include different filter [6] and optimization [7] algorithms. While all these algorithms are well suited for real robotic applications, none of them takes into account how the motion of the robot can be exploited to improve the accuracy of the estimation, both in terms of quality and speed, i.e., they do not *actively* control the robot.

Active reconstruction algorithms seek to optimize an objective function that represents the quality of the reconstruction, which is often formulated as choosing the control action that minimizes the error covariance [8] or maximizes the expected information gain [9]. Next-best-view (NBV) planning selects the next camera position from a discrete action set, also in the interest of minimizing the covariance [10] or maximizing the information gain [11]. Similarly, higher level camera motions can be planned to verify a hypothesis about the object's shape [12], determine a set of new camera views that improve an existing 3-D mesh [13], or maximize the information gain with respect to a probabilistic depth map [14]. Mostegel *et al.* [15] generate a probabilistic obstacle map that a quadrotor uses to actively explore an unknown scene by evaluating a heuristic composed of view angle maximization and probability of point recognition. In terms of high-speed aerial robotics, Barry and Tedrake [16] use a stereo-vision, block-matching algorithm to detect and avoid obstacles. Unfortunately, many of these active vision-based approaches require discretization of the space, the solution of an optimization problem in-the-loop, or lack probabilistic models, making them difficult to apply in a dynamic, reactive setting, with limited computational resources, e.g., on board an agile, maneuvering aerial robot.

Control theoretic approaches for active sensing have also been applied to the task of 3-D reconstruction. Spica and Giordano [17] proposed an active SfM observer and control strategy for a general class of nonlinear systems that actively tune the transient response of the error dynamics. They apply their method to active 3-D reconstruction of feature points—as in our work—with a six degree of freedom (DOF) mobile camera. This paper was extended later for active estimation of the pose for specific object types such as point, sphere, and cylinder objects [18] and, later, planar scenes [19], [20] as well. Salaris *et al.* [21] proposed an optimal sensing strategy that actively maximizes the smallest eigenvalue of the observability Gramian matrix for differentially flat systems with noisy measurements. The online optimal control solution estimates the state via an EKF and actively maneuvers the control points of B-splines in a gradient descent fashion. This method was recently extended by Cognetti *et al.* [22] to systems with process noise on the dynamics by instead minimizing the largest eigenvalue of the covariance matrix from solving the continuous Riccati equation and then similarly choosing spline control points via gradient descent.

Both our work and this active SfM literature have similar motivation in that we desire a control input that lowers the uncertainty in the system. However, the active SfM optimization problem is generally to maximize a function of the Fisher information matrix while we choose to consider lowering a function of the covariance matrix. Conceptually, they are similar but provide slightly different performance as we show in our results. The most important difference is that our method plans in a Bayesian sense with full probability distributions and is dynamical system agnostic (e.g., not constrained to linearly coupled unobservable states [17] or differentially flat systems [21], [22]).

The control community has also considered driving systems to states that minimize some objective in real time. For example, extremum seeking control (ESC) is an approach that seeks to optimize a measurable target function online. Krstic and Wang [23] provided the first stability analysis of an extremum seeking feedback control for single-input single-output systems driven in a gradient descent manner. Moase *et al.* [24] proposed using a Netwon-like optimization to improve convergence performance. Recent approaches consider simplex guided extremum seeking (SGES) that follow a gradient-based approach (i.e., ESC) to a local extremum and further employ either multidirectional search algorithms [25], or, in later work, a hybrid optimization algorithm [26] to find global extremum. In fact, these methods have been applied to visual eye-in-hand searches for objects of interest [27]. However, ESC is historically sensitive to measurement noise and often does not consider estimation in a Bayesian framework as we do in this paper.

Results from optimal sensor placement literature also provide key insights into the active vision problem. Olague and Mohr [28] perform a global optimization for camera network design by minimizing the worst case eigenvalue of a covariance matrix. Cowen and Kovesi [29] describe a heuristic for this problem that selects the best camera placement that has the most region of overlap considering satisfying tasks. Although promising, the bulk of active sensor placement literature is not aimed at fast, online control methods for mobile cameras and is often restricted to discrete domains for computational simplicity.

To conclude, our work is most similar in concept to the target tracking literature, which utilizes filtering and focuses on minimizing a function of the estimation error covariance matrix. Yang *et al.* [30] propose a square-root unscented strong tracking filter for target tracking and estimation in asynchronous networks. Zhou and Roumeliotis [31] propose to solve for a set of feasible sensor locations that minimize the uncertainty of a tracked target. In the multirobot setting, Campbell and Whitacre [32] consider the cooperative tracking using a square-root sigma point information filter while

Vela *et al.* [33] consider using vision to enable leader-follower formations. Morbidi and Mariottini [34] solve the multivehicle target tracking problem for a linear system with the second-order vehicle dynamics and choose the acceleration control input equal to either the gradient decent of trace, determinant, or maximum eigenvalue of the fused covariance matrix with an additive damping term. Finally, Wei *et al.* [35] perform target tracking for a nonlinear system of fused target and tracker states and move the trackers by computing the gradient of future covariance with respect to the control input. In our work, the major difference is that we seek to estimate the 3-D structure of an environment rather than tracking positions of targets with a (typically) known camera position. Furthermore, we employ a realistic camera sensing model and quadrotor kinematic model to the active estimation problem.

II. Problem Formulation

We first formalize the general active sensing problem. Consider a stochastic, potentially nonlinear, dynamical system with unknown state $\mathbf{x}_k \in \mathbb{R}^n$ over discrete, nonnegative time steps $k \in \mathbb{Z}_+$, with deterministic control inputs $\mathbf{u}_k \in \mathbb{U}_k \subset \mathbb{R}^p$, where $\mathbb{U}_k$ is the allowable control space. We use the notation $\mathbf{u}_{0:k}$ to denote a sequence of control inputs $(\mathbf{u}_0, \ldots, \mathbf{u}_k)$. The system evolves according to the known Markov dynamics $p(\mathbf{x}_{k+1}) = \int_{\mathbf{x}_k} p(\mathbf{x}_{k+1} \mid \mathbf{x}_k; \mathbf{u}_k) p(\mathbf{x}_k)\, d\mathbf{x}_k$, where we use the notation $(\cdot\,; \mathbf{u}_k)$ to indicate that $\mathbf{u}_k$ is a deterministic parameter of the state transition distribution. We take measurements $\mathbf{y}_k$ at time k, taken from a known sensing distribution $p(\mathbf{y}_k \mid \mathbf{x}_k)$. Let $\mathrm{p}_k := p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ denote the posterior estimate of the state given all measurements up to the current time. We compute p_k with the recursive Bayesian filter

$$\mathrm{p}_k = \eta_k\, p(\mathbf{y}_k \mid \mathbf{x}_k) \int_{x_{k-1}} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}; \mathbf{u}_{k-1}) \mathrm{p}_{k-1}\, d\mathbf{x}_{k-1} \quad (1)$$

where η_k is a normalization constant ensuring the distribution integrates to 1. Active sensing, in our context, refers to finding a control policy $\mathbf{u}_k = \gamma_k(\mathrm{p}_k)$ so as to minimize the *expected uncertainty* in the state estimate p_{k_f} over some time horizon k_f, where k_f may be infinite. [1]

Let $J(\mathrm{p}_{k_f})$ be the cost function that we wish to optimize. Common choices for this function are the Shannon entropy, or the trace, determinant, or maximum eigenvalue of the covariance matrix of the state estimate vector. We can phrase the problem of active sensing as the following:

$$\min_{\mathbf{u}_{0:k_f}} \mathop{\mathbb{E}}_{\mathbf{y}_{1:k_f}} [J(\mathrm{p}_{k_f})]$$

$$\text{s.t. } \mathrm{p_k} = \eta_k \mathrm{p}(\mathbf{y_k}|\mathbf{x_k}) \int_{\mathbf{x}_{k-1}} \mathrm{p}(\mathbf{x_k} \mid \mathbf{x}_{k-1}; \mathbf{u}_{k-1}) \mathrm{p}_{k-1}\, \mathrm{d}\mathbf{x}_{k-1}. \quad (2)$$

[1] Here, we have chosen for the control input to only affect the state transition $p(\mathbf{x}_{k+1} \mid \mathbf{x}_k; \mathbf{u}_k)$, but in general, it may affect the sensing function $p(\mathbf{y}_k \mid \mathbf{x}_k; \mathbf{u}_k)$, as well. We focus on controlling the state transition in this paper, as this is the relevant model for our specific 3-D vision active estimation application.

The well-known solution to this Dynamic program via Bellman's backward recursion is then given by

$$J_k(\mathrm{p}_k) = \min_{\mathbf{u}_k} \mathop{\mathbb{E}}_{\mathbf{y}_{k+1}} [J_{k+1}(\mathrm{p}_{k+1}) \mid \mathbf{y}_{1:k}]$$
$$\text{s.t. } \mathrm{p}_{k+1} = \eta_{k+1} \mathrm{p}(\mathbf{y}_{k+1}|\mathbf{x}_{k+1}) \int_{\mathbf{x}_k} p(\mathbf{x}_{k+1}|\mathbf{x}_k; \mathbf{u}_k) \mathrm{p}_k\, d\mathbf{x}_k \quad (3)$$

with the corresponding optimal control policy $\gamma_k(\mathrm{p}_k) = \arg\min_{\mathbf{u}_k} \mathop{\mathbb{E}}_{\mathbf{y}_{k+1}} [J_{k+1}(\mathrm{p}_{k+1}) \mid \mathbf{y}_{1:k}]$, where $J_k(\cdot)$ is the cost-to-go function at time k.

Unfortunately, obtaining the optimal solution to this dynamic program is known to be computationally expensive, and is often intractable. This is because the integral and minimization in (3) are often impossible to evaluate analytically, so numerical approximations must be used. Problems of this type are sometimes formulated as partially observable Markov decision processes (POMDPs), where the solution is a control policy, mapping from points in the belief space to an optimal action. However, active sensing problems do not directly fit this framework because the objective is a function of the belief, not only the state and action. Therefore, all planning must be done in the belief space which often requires significant discretization or sampling. Moreover, modern online methods are simply too computationally demanding for MAVs [36].

Many approaches to active sensing attempt to approximately solve this dynamic program. These include NBV approximations, which greedily seek to minimize $\mathbb{E}_{\mathbf{y}_{k+1}}[J(\mathrm{p}_{k+1}) \mid \mathbf{y}_{1:k}]$, or receding horizon solutions that attempt to solve a problem of the form (3) for a short time horizon at each step. Instead, we propose a gradient descent approach to approximately solve (2), which is similar in concept to (3), but much cheaper in terms of computational demands, and provides acceptable results in terms of quality of the estimation.

First, we denote the mean $\hat{\mathbf{x}}_k := \mathbb{E}[x_k \mid \mathbf{y}_{1:k}]$, and predicted mean $\hat{\mathbf{x}}^-_{k+1} := \mathbb{E}[x_{k+1} \mid \mathbf{y}_{1:k}]$, and we can express the predicted mean with the dynamical equation $\hat{\mathbf{x}}^-_{k+1} = f(\hat{\mathbf{x}}_k, \mathbf{u}_k)$, where $f(\hat{\mathbf{x}}_k, \mathbf{u}_k) := \int_{\mathbf{x}_k} p(\mathbf{x}_{k+1} \mid \mathbf{x}_k; \mathbf{u}_k) \mathrm{p}_k\, d\mathbf{x}_k$. We propose to find the control $\mathbf{u}_k$ at each step to solve the equation for the one-step gradient descent update rule

$$\hat{\mathbf{x}}^-_{k+1} = \hat{\mathbf{x}}_k - \Gamma \nabla_{\hat{\mathbf{x}}_k} \mathbb{E}_{\mathbf{y}_{k+1}}[J(\mathrm{p}_{k+1}) \mid \mathbf{y}_{1:k}]$$

with a constant control gain, Γ. This algorithm will drive the state estimate in the expected direction of steepest descent with respect to the cost function. Furthermore, we choose the trace of the covariance matrix as the cost function in this paper, $J(\mathrm{p}_k) = \mathrm{Tr}(\Sigma_k)$, where $\Sigma_k := \mathbb{E}[(\mathbf{x} - \hat{\mathbf{x}}_k)(\mathbf{x} - \hat{\mathbf{x}}_k)^T \mid \mathbf{y}_{1:k}]$. Since the trace is linear, we can move the gradient operator inside the trace. Finally, the goal of our active sensing control is to solve the following equation for $\mathbf{u}_k$:

$$f(\hat{\mathbf{x}}_k, \mathbf{u}_k) = \hat{\mathbf{x}}_k - \Gamma \mathrm{Tr}(\nabla_{\hat{\mathbf{x}}_k} \Sigma_{k+1}) \quad (4)$$

given the dynamics function $f(\cdot, \cdot)$. The resulting action is the control input that *drives* our estimated state down the gradient of the estimation error covariance matrix; conceptually similar methods have been called "information surfing" in the past.

Remark 1 (Estimated State): While gradient descent has been extensively used in the active sensing literature, there are

two key differences in our setup, which make its discussion worthwhile. First, we do not have access to the true state, $\mathbf{x}_k$, so we cannot move the sensor according to the true gradient. Consequently, our strategy moves in the best possible direction with the best of current knowledge, which will only be as good as the quality of the estimation, $\hat{\mathbf{x}}_k$. Second, because we consider controlling a dynamical process in order to better estimate its state, we have to solve the dynamics equation in (4) to find the appropriate control action.

In Section III, we derive the special form of this control algorithm for the cases when the state covariance matrix in (4) is approximated with either the EKF or the UKF.

III. Active Sensing With the EKF and UKF

In this section, we first briefly review the EKF and UKF algorithms to include the necessary notation to follow the active sensing control. We then derive the gradient of the one-step prediction covariance matrix with respect to the mean state to be used in the active sensing algorithm described in the previous section. We compute the gradients with respect to each mean state estimate, indexed by j, thus we are interested in computing $\boldsymbol{\nabla}_j := (\partial)/(\partial \hat{\mathbf{x}}_j)\mathrm{Tr}(\Sigma)$, where $\boldsymbol{\nabla} := \nabla_{\hat{\mathbf{x}}}\mathrm{Tr}(\Sigma)$.

A. Active Extended Kalman Filtering

The EKF applies the typical Kalman filter updates to a linearized model of a nonlinear system. The approximation estimates linear system matrices, $\mathbf{F}_k$ and $\mathbf{H}_k$, by evaluating the first term from the Taylor series expansion of the dynamics function, $f(\cdot)$, and the measurement function, $h(\cdot)$, respectively, around the current state estimate. This requires evaluating the Jacobian of each of these functions. These approximations are then used in the standard Kalman filter's prediction and update strategy to yield a state estimate

$$\begin{aligned}
\hat{\mathbf{x}}_{k+1}^{-} &= f(\hat{\mathbf{x}}_k, \mathbf{u}_k) \\
\Sigma_{k+1}^{-} &= \mathbf{F}_{k+1}\Sigma_k\mathbf{F}_{k+1}^T + \mathbf{Q} \\
\hat{\mathbf{x}}_{k+1} &= \hat{\mathbf{x}}_{k+1}^{-} + \mathbf{G}_{k+1}\big(\mathbf{y}_{k+1} - h\big(\hat{\mathbf{x}}_{k+1}^{-}\big)\big) \\
\Sigma_{k+1} &= \Sigma_{k+1}^{-} - \mathbf{G}_{k+1}\mathbf{H}_{k+1}\Sigma_{k+1}^{-}
\end{aligned} \tag{5}$$

with Kalman gain $\mathbf{G}_{k+1} = \Sigma_{k+1}^{-}\mathbf{H}_{k+1}^T(\mathbf{H}_{k+1}\Sigma_{k+1}^{-}\mathbf{H}_{k+1}^T + \mathbf{R})^{-1}$.

Active extended Kalman filtering via gradient descent computes a control input at time k, $\mathbf{u}_k$, from (4) where the state estimate is provided by the EKF. We state the following proposition that describes the structure of the gradient in terms of matrices computed in the EKF and the gradient of the linearized matrices, $\mathbf{F}$ and $\mathbf{H}$. For the sake of clarity, we omit the temporal dependencies on our notation in the following propositions, e.g., Σ_{k+1} becomes Σ.

Proposition 1 (Active EKF Gradient): The jth component of the gradient vector for the EKF is defined by

$$\boldsymbol{\nabla}_j = \mathrm{Tr}\left((\mathbf{I}_n - 2\mathbf{G}\mathbf{H})2\mathbf{F}\Sigma\frac{\partial \mathbf{F}^T}{\partial \hat{x}_j} - 2\mathbf{G}\frac{\partial \mathbf{H}}{\partial \hat{x}_j}\Sigma^{-} + \mathbf{G}\frac{\partial \mathbf{M}}{\partial \hat{x}_j}\mathbf{G}^T\right) \tag{6}$$

where

$$\frac{\partial \mathbf{M}}{\partial \hat{x}_j} = 2\mathbf{H}\Sigma^{-}\frac{\partial \mathbf{H}^T}{\partial \hat{x}_j} + 2\mathbf{H}\mathbf{F}\Sigma\frac{\partial \mathbf{F}^T}{\partial \hat{x}_j}\mathbf{H}^T. \tag{7}$$

Proof: See Appendix VI-A for the derivation of this result. □

Therefore, the gradient can be computed by multiplying matrices already available from the EKF equations (5). The only required elements are the gradients of $\mathbf{F}$ and $\mathbf{H}$, which are

$$\frac{\partial \mathbf{F}}{\partial \hat{x}_j} = \frac{\partial}{\partial \hat{x}_j}\left(\frac{\partial f}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}_k, \mathbf{u}_k}\right) \tag{8}$$

and

$$\frac{\partial \mathbf{H}}{\partial \hat{x}_j} = \frac{\partial}{\partial \hat{x}_j}\left(\frac{\partial h}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}_{k+1}^{-}}\right) \tag{9}$$

respectively.

B. Active Unscented Kalman Filtering

The UKF is a linearization technique that deterministically samples *sigma points* from the prior distribution and propagates these few points through the true nonlinear system rather than a linearized version.[2] The prior and posterior estimates are given by

$$\begin{aligned}
\hat{\mathbf{x}}_{k+1}^{-} &= \boldsymbol{\chi}_{k+1}^{-}\boldsymbol{w} \\
\Sigma_{k+1}^{-} &= \big[\mathbf{1}_N \otimes \hat{\mathbf{x}}_{k+1}^{-} - \boldsymbol{\chi}_{k+1}^{-}\big] \\
&\quad \times \mathbf{W}\big[\mathbf{1}_N \otimes \hat{\mathbf{x}}_{k+1}^{-} - \boldsymbol{\chi}_{k+1}^{-}\big]^T + \mathbf{Q} \\
\hat{\mathbf{x}}_{k+1} &= \hat{\mathbf{x}}_{k+1}^{-} + \mathbf{G}_{k+1}\big(\mathbf{y}_{k+1} - \hat{\mathbf{y}}_{k+1}^{-}\big) \\
\Sigma_{k+1} &= \Sigma_{k+1}^{-} - \mathbf{G}_{k+1}\Sigma_{\mathbf{y}_{k+1},\mathbf{y}_{k+1}}\mathbf{G}_{k+1}^T
\end{aligned} \tag{10}$$

where $\otimes$ represents the *Kronecker product*. The difference from the EKF is the use of the *sigma point matrix*, $\boldsymbol{\chi}_k = [\chi_k^{[0]}, \ldots, \chi_k^{[2n]}] \in N \times \mathbb{R}^n$, which contains the $N = (2n+1)$ sampled sigma points. This matrix is populated using the posterior state estimate and covariance matrix as

$$\boldsymbol{\chi}_k = [\hat{\mathbf{x}}_k,\ \mathbf{1}_n \otimes \hat{\mathbf{x}}_k + \sqrt{\lambda\Sigma_k},\quad \mathbf{1}_n \otimes \hat{\mathbf{x}}_k - \sqrt{\lambda\Sigma_k}] \tag{11}$$

where $\mathbf{1}_n = [1, \ldots, 1]^T \in \mathbb{R}^n$ and λ is a scale factor. Associated with each sigma point is a constant weight for both the mean and covariance that is used to compute the new estimates via weighted summation. Here, we denote the vector of weights for the mean by $\boldsymbol{w} = [w_m^{[0]}, \ldots, w_m^{[2n]}]^T$ and the diagonal matrix of weights for the covariance by $\mathbf{W} = \mathrm{diag}([w_c^{[0]}, \ldots, w_c^{[2n]}]^T)$. The weights are chosen according to [37].

At each time step, the sigma points are generated according to (11) and are propagated through the nonlinear dynamics, yielding the predicted sigma points

$$\boldsymbol{\chi}_{k+1}^{-} = \big[f\big(\chi_k^{[0]}, \mathbf{u}_k\big), \ldots, f\big(\chi_k^{[2n]}, \mathbf{u}_k\big)\big]. \tag{12}$$

The sigma points are similarly propagated through the measurement nonlinearity and used to populate the matrix $\boldsymbol{y}_{k+1}^{-} \in \mathbb{R}^{m\times N}$ in a similar fashion as (12), only replacing $f(\chi_k^{[i]}, \mathbf{u}_k)$

[2]In principal, the UKF could be replaced with the Particle Filter where particles are instead randomly sampled from state space and then propagated through the nonlinear system. We do not consider the Particle Filter in this analysis because we have found in simulations that it requires far more computation than the other two filters, making it less practical for fast control.

with $h(\chi_{k+1}^{[i]-})$. Finally, the update in (10) is performed by calculating the following additional matrices:

$$\begin{aligned}\hat{\mathbf{y}}_{k+1}^{-} &= \boldsymbol{y}_{k+1}^{-}\boldsymbol{w} \\ \Sigma_{\mathbf{x}_{k+1},\mathbf{y}_{k+1}} &= \left[\mathbf{1}_N \otimes \hat{\mathbf{x}}_{k+1}^{-} - \boldsymbol{\chi}_{k+1}^{-}\right]\mathbf{W} \\ &\quad \left[\mathbf{1}_N \otimes \hat{\mathbf{y}}_{k+1}^{-} - \boldsymbol{y}_{k+1}^{-})\right]^T \\ \Sigma_{\mathbf{y}_{k+1},\mathbf{y}_{k+1}} &= \left[\mathbf{1}_N \otimes \hat{\mathbf{y}}_{k+1}^{-} - \boldsymbol{y}_{k+1}^{-}\right]\mathbf{W} \\ &\quad \left[\mathbf{1}_N \otimes \hat{\mathbf{y}}_{k+1}^{-} - \boldsymbol{y}_{k+1}^{-}\right]^T + \mathbf{R} \\ \mathbf{G}_{k+1} &= \Sigma_{\mathbf{x}_{k+1},\mathbf{y}_{k+1}} \Sigma_{\mathbf{y}_{k+1},\mathbf{y}_{k+1}}^{-1}. \end{aligned} \tag{13}$$

Active uscented Kalman filtering via gradient descent also computes a control input at time k, $\mathbf{u}_k$, from (4) where the state estimate is now provided by the UKF. We state the following proposition that describes the structure of the gradient.

Proposition 2 (Active UKF Gradient): The jth component of the gradient vector for the UKF is defined by

$$\boldsymbol{\nabla}_j = \mathrm{Tr}\left(\frac{\partial \Sigma^{-}}{\partial \hat{x}_j} - 2\mathbf{G}\frac{\partial \Sigma_{\mathbf{x},\mathbf{y}}^{T}}{\partial \hat{x}_j} + \mathbf{G}\frac{\partial \Sigma_{\mathbf{y},\mathbf{y}}}{\partial \hat{x}_j}\mathbf{G}^T\right). \tag{14}$$

The partial derivatives of each covariance matrix are weighted sums of other partial derivatives. In fact, the only system specific required derivates are

$$\frac{\partial f\left(\chi_k^{[i]}, \mathbf{u}_k\right)}{\partial \hat{x}_j} = \frac{\partial f\left(\chi_k^{[i]}\right)}{\partial \chi_{k,j}^{[i]}}\frac{\partial \chi_{k,j}^{[i]}}{\partial \hat{x}_j} \tag{15}$$

and

$$\frac{\partial h\left(\chi_{k+1}^{[i]-}\right)}{\partial \hat{x}_j} = \frac{\partial h\left(\chi_{k+1}^{[i]-}\right)}{\partial \chi_{k+1,j}^{[i]-}}\frac{\partial f\left(\chi_k^{[i]}, \mathbf{u}_k\right)}{\partial \hat{x}_j} \tag{16}$$

where

$$\frac{\partial \chi_{k,j}^{[i]}}{\partial \hat{x}_j} = 1. \tag{17}$$

Proof: See Appendix VI-B for the derivation of this result. □

C. Active Filtering Discussion

We would like to point out a few interesting observations regarding the previously defined gradients. First, the partial derivatives for the EKF gradient are actually partials of the linearized dynamics and measurement model, as shown in [34] and [38]. This is not the case for the UKF, since it propagates the sigma points using the full nonlinear system. This is an important difference since the partial derivatives of the estimate will not be entirely accurate in the active EKF setting.

Regarding computational demand, the structure of partial derivatives of the nonlinear system (or linearization matrices for the EKF) may be determined offline. These matrices must be evaluated at each time step with the current state estimate, however, this is generally computationally inexpensive. The subsequent control includes the simple matrix multiplications and a final trace operation for each component in the state vector. We have found that modern processing hardware can handle these computations in real time with no problems. The UKF is known to be on the order of $2n$ times more computationally expensive due to the additional $2n$ states, but does not rely on the linearization matrices as the EKF does, making the partial derivatives more accurate.

Finally, we observe that the desired control input $\mathbf{u}_k$ appears in the partial derivatives in (8), (15), and (16). Depending on the structure of $f(\mathrm{x}_k, \mathbf{u}_k)$, the solution for the control input may prove difficult. Fortunately, in our active 3-D reconstruction problem, these partials are simple to compute and are actually not a function of $\mathbf{u}_k$ as the dynamics are affine (more details are provided in the following section).

IV. Vision-Based Active Sensing For an Aerial Robot

In the rest of this paper, we apply and demonstrate the active EKF and active UKF control with a 3-D vision example using an airborne camera on a quadrotor. The state here represents the Euclidean position of M features in an image with respect to the current camera's coordinate system. In this way, the structure in the scene is robot-centric and not a global map. For clarity, we introduce the system and active sensing control for one image feature, i.e., $M = 1$, and add the extension to multiple features afterward. The state vector is then $\mathbf{x} = [x_x, x_y, x_z]^T \in \mathbb{R}^3$ and the measurement vector is $\mathbf{y} = [y_x, y_y]^T \in \mathbb{R}^2$. The state size is, therefore, $n = 3$.

A. Kinematic Camera Control

The system's motion is described by the continuous-time kinematics for one 3-D feature with respect to a mobile coordinate frame in SE(3)

$$\dot{\mathbf{x}} = -\boldsymbol{\Omega}\mathbf{x} - \boldsymbol{v} \tag{18}$$

where $\boldsymbol{\Omega} = \boldsymbol{\omega}^{\wedge}$ is the skew symmetric matrix form of the vector $\boldsymbol{\omega} \in \mathbb{R}^3$ that defines the body-frame rotational rates and $\boldsymbol{v} \in \mathbb{R}^3$ is the vector of translational velocities.

Our algorithm computes the desired translational camera velocities at each time step while the quadrotor simultaneously tracks this desired body-frame velocity to produce the final active camera motion. We select the translational velocity and not the rotation rates ($\boldsymbol{\omega}_k$) for two reasons. The first and most important stems from the SfM literature, where it is well understood that pure camera rotations will not aid in refining the unknown depth of a feature. This can be shown from the epipolar constraint, given as $\mathbf{x}_{k+1}^T(\mathbf{t}^{\wedge}\boldsymbol{R})\mathbf{x}_k = 0$ (see [2, eq. (5.2)]), where a null $\mathbf{t}$ vector results in the trivial solution for the epipolar constraint regardless of the rotation matrix $\boldsymbol{R} \in SO(3)$, thus providing no new information about the relative transformation between image frames.

The second reason is one of the implementations. Most unmanned aerial vehicles (UAVs) have coupled translational and rotational velocities, thus we cannot choose to control both independently. When considering the quadrotor, it is clear that (18) does not contain the vehicle dynamics. However, (18) does represent the kinematic equations for a mobile coordinate frame in SE(3)—including a quadrotor—and is used to keep the motion general in terms of vehicle type, allowing for future substitution of other dynamical systems that perform

velocity tracking, e.g., a ground robot, robot arm, or underwater autonomous vehicle. Nevertheless, this model is often used when considering a mobile camera on an aerial vehicle (see [17], [39]).

Alternatively, one could consider including the dynamics of the 3-D feature points in (18) and selecting the best torques and forces as the direct, low-level control input. We do not do this because noisy motor control signals can cause unstable flight, leading to dangerous quadrotor crashes. This technique would also require extracting the control input from the nonlinear dynamics, leading to increased computational demand. Most of all, trajectory or velocity tracking is generally regarded as a solved problem and there exists a wide variety of quadrotor control methods that produce smooth motions from desired velocities [40].

We only consider the motion of the quadrotor on a plane above the object of interest as an operational constraint. The large depth uncertainty in the first few time-steps of the algorithm causes the active inputs to drive the camera directly toward the object of interest with the intention of increasing parallax between consecutive image frames. We, therefore, restrict this motion to reduce the risk of collision and to keep the scene sufficiently broad in the image, ensuring the camera has the opportunity to track the features long enough to refine the 3-D position estimate. Finally, we control the quadrotor to track desired body-frame velocities using a standard quadrotor control framework [41].

Remark 2 (Coincident Coordinate Systems): We treat the camera coordinate frame as coincident with the quadrotor's inertial measurement frame in this paper. Although this is not true, in practice, the body-frame input velocities can be easily transformed into the quadrotor frame given the known, static relative camera pose with respect to the quadrotor's frame. We handle the full transformation in our experiments.

Assumption 1 (Body Velocities and Euler Angles): We finally assume that we have an estimate of the quardotor's angular rotation rates, the translational velocities, and Euler angles for the subsequent low-level quadrotor control. The Euler angles and rotation rates are easily extracted from the on-board inertial measurement unit (IMU) with some filtering if necessary. The translational velocity is difficult to obtain from pure IMU data, but we assume it may be estimated via the on-board state estimator or by using visual odometry methods [42], [43].

B. Quadrotor Model and Control

The quadrotor's standard nonlinear dynamical equations [44] are

$$\begin{aligned}\dot{\mathbf{r}}_w &= \boldsymbol{v}_w \\ \dot{\boldsymbol{v}}_w &= \mathbf{g} + \frac{1}{m}\boldsymbol{R}_{wr} F e_3 \\ \dot{\boldsymbol{R}}_{wr} &= \boldsymbol{R}_{wr}\Omega \\ \dot{\boldsymbol{\omega}} &= \mathbf{J}^{-1}\boldsymbol{\tau} - \mathbf{J}^{-1}\Omega\,\mathbf{J}\boldsymbol{\omega}. \end{aligned} \tag{19}$$

In this model, the robot's position in some fixed world frame is denoted by $\mathbf{r}_w$, $\boldsymbol{v}_w$ is the velocity applied in this world frame, the gravity vector in the world frame is denoted by $\mathbf{g} = [0, 0, 9.81]^T \mathrm{m/s}^2$, m is the mass of the quadrotor measured in kg, $\boldsymbol{R}_{wr}$ is the rotation from the world coordinate frame to the robot's local body frame, F is the total applied thrust force scalar that is aligned with the quadrotor's body-frame z-axis (denoted by the third canonical basis vector $e_3 = [0, 0, 1]^T$), $\boldsymbol{\Omega}$ is defined as the skew symmetric matrix of the rotation rates in the same manner as (18), $\boldsymbol{\omega} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ is the vector of body-frame rotation rates, $\mathbf{J}$ is the moment of inertia matrix, and finally, $\boldsymbol{\tau}$ is the vector of applied torques.

We control the quadrotor given the calculated active perception body frame, desired translational velocity, defined by $\mathbf{u}_k^{\mathrm{des}}$, in a similar fashion to the controllers presented in [40]. Given $\mathbf{u}_k^{\mathrm{des}}$, we determine the desired Euler angles with the velocity controller and then pass these into a separate attitude controller that tracks these Euler angles. The velocity proportional—integral—differential (PID) controller is a function of the velocity error, which outputs desired body-frame accelerations for the quadrotor

$$\dot{\boldsymbol{v}}_k^{\mathrm{des}} = K_p e_{\boldsymbol{v}_k} + \int K_i (e_{\boldsymbol{v}_k} + e_{\boldsymbol{v}_{k-1}})\, dt + K_d \frac{d}{dt}(e_{\boldsymbol{v}_k} - e_{\boldsymbol{v}_{k-1}}) \tag{20}$$

where $e_{\boldsymbol{v}_k} = \boldsymbol{v}_k - \mathbf{u}_k^{\mathrm{des}}$, $K_p \leq 0$, $K_i \leq 0$, and $K_d \leq 0$ are the proportional, integral, and derivative gains, respectively. The gains are the diagonal matrices where the diagonal elements represent the gains for each direction, i.e., x, y, z. The altitude of the quadrotor is controlled to the initial condition using the total input force: $F = -m(\mathbf{g})/(cos(\phi)cos(\theta))$. This controller is sufficient when the Euler angles remain small as in this case.

The desired Euler angles are then defined by

$$\begin{aligned}\phi^{\mathrm{des}} &= \frac{1}{g}\left(\dot{\boldsymbol{v}}_{k,x}^{\mathrm{des}} - \dot{\boldsymbol{v}}_{k,y}^{\mathrm{des}}\right) \\ \theta^{\mathrm{des}} &= \frac{1}{g}\left(\dot{\boldsymbol{v}}_{k,x}^{\mathrm{des}} + \dot{\boldsymbol{v}}_{k,y}^{\mathrm{des}}\right) \end{aligned} \tag{21}$$

where $g = 9.81\ \mathrm{m/s}^2$. This controller is different from the one presented by Mellinger and Kumar [41] because it only requires desired planar, body-frame quadrotor velocities and does not rely on the generation of a desired trajectory in the world frame. Finally, we are not concerned with the yaw of the camera so we omit the yaw angle control.

The input torques are calculated with the following PD controller:

$$\begin{aligned}\tau_{k,x} &= K_{p,\phi}(\phi_k - \phi_k^{\mathrm{des}}) + K_{d,\phi}(\boldsymbol{\omega}_{k,x} - \boldsymbol{\omega}_{k,x}^{\mathrm{des}}) \\ \tau_{k,y} &= K_{p,\theta}(\theta_k - \theta_k^{\mathrm{des}}) + K_{d,\theta}(\boldsymbol{\omega}_{k,y} - \boldsymbol{\omega}_{k,y}^{\mathrm{des}}), \end{aligned} \tag{22}$$

where $K_{p,\phi} \leq 0$ and $K_{p,\theta} \leq 0$ are the proportional control gains, whereas $K_{d,\phi} \leq 0$ and $K_{d,\theta} \leq 0$ are the derivative gains for motion about the x and y axes, respectively.

C. Measurement Model

The measurement model is defined by the pinhole camera model [2]. This model describes the transformation from a 3-D world feature, $\mathbf{x}$, to the projected feature point in the camera's image, $\mathbf{y}$

$$\mathbf{y} = \frac{1}{x_z}\begin{bmatrix}\alpha_x & 0 & p_{x_0} \\ 0 & \alpha_y & p_{y_0}\end{bmatrix}\begin{bmatrix}x_x \\ x_y \\ x_z\end{bmatrix} = \mathbf{K}\begin{bmatrix}\frac{x_x}{x_z} \\ \frac{x_y}{x_z} \\ 1\end{bmatrix} = h(\mathbf{x}) \tag{23}$$

where α_x and α_y are the horizontal and vertical scale factors that relate world length units to pixels, $[p_{x_0}, p_{y_0}]^T$ is the principal point of the sensor (i.e., the pixel location where the optical axis is orthogonal to the imaging plane), and $\mathbf{K} \in \mathbb{R}^{2\times 3}$ is the known camera calibration matrix that is determined offline.

D. Active 3-D Vision System

The final nonlinear system for feature l at time k can be summarized by

$$\mathbf{x}_{k+1} = e^{-\Omega_k \Delta t}\mathbf{x}_k - \mathbf{u}_k \Delta t + \mathbf{v} \tag{24a}$$

$$\mathbf{y}_k = \frac{1}{x_z}\mathbf{K}\mathbf{x}_k + \mathbf{w} \tag{24b}$$

where Δt is the step size, $\mathbf{x}_k = [x_{k,x}, x_{k,y}, x_{k,z}]^T \in \mathbb{R}^3$. Gaussian white noise vectors, $\mathbf{v} \sim \mathcal{N}(\mathbf{0}_3, \mathbf{Q} \in \mathbb{R}^{3\times 3})$ and $\mathbf{w} \sim \mathcal{N}(\mathbf{0}_2, \mathbf{R} \in \mathbb{R}^{2\times 2})$, describe the process and measurement noise, respectively, and $\mathbf{0}_n = [0, \ldots, 0]^T \in \mathbb{R}^n$. Note that the feature kinematics (24a) is linear in the state $\mathbf{x}_k$. In fact, the kinematics represent the rigid body transformation from the coordinate system at time k to time $k+1$, where $e^{-\Omega_k \Delta t} \in$ SO(3) is the active rotation matrix from frame $k+1$ to k and $(-\mathbf{u}_k \Delta t + \mathbf{v})$ is the translation vector. The linear kinematics allows the Bayesian prediction to remain identical to the prediction step from the Kalman filter. However, the linear Kalman filter update may not be performed due to the nonlinear measurement model (24b).

E. Active Perception Control

Considering again the trace of Σ_{k+1} as the cost function to minimize at each step, combining (4) and (24) we obtain

$$(e^{-\Omega_k \Delta t} - \mathbf{I}_n)\hat{\mathbf{x}}_k - \mathbf{u}_k \Delta t = -\mathrm{Tr}(\nabla_{\hat{\mathbf{x}}} \Sigma_{k+1}). \tag{25}$$

Solving for the control input at time k results in

$$\mathbf{u}_k = (e^{-\Omega_k \Delta t} - \mathbf{I}_n)\hat{\mathbf{x}}_k + \nabla \tag{26}$$

where $\mathbf{I}_n = \mathrm{diag}(\mathbf{1}_n)$ and the vector ∇ has components $\nabla_j = \partial/\partial \hat{x}_j \mathrm{Tr}(\Sigma_{k+1})$, where $\gamma_j \in \{\nabla_x, \nabla_y, \nabla_z\}$ and $\hat{x}_j \in \{\hat{x}_x, \hat{x}_y, \hat{x}_z\}$ for the single feature case. Note that only the partial derivatives with respect to the planar feature position are necessary to drive the robot on a plane. Thus, the third partial, ∇_z is not used; however, we present this form of the partial for generality. We have removed the resulting $1/\Delta t$ from the control input as this scalar may be included in a final applied gain.

The final active perception control input applied on the robot follows the same final scaling for all 3-D feature cases

$$\mathbf{u}_k^{\mathrm{des}} := \Gamma \frac{\mathbf{u}_k}{\|\mathbf{u}_k\| + \epsilon}. \tag{27}$$

This desired translational velocity is defined by a *soft saturation* of the computed control, $\mathbf{u}_k$. This saturation perserves the direction of $\mathbf{u}_k$, but constrains the magnitude of the velocity to a maximum value defined by the gain matrix, $\Gamma = v^{\mathrm{des}}\mathrm{diag}([1, 1, 0]^T)$. Furthermore, a small value ϵ is added to the denominator to prevent singularities when the control action approaches zero. Constraining the control in this manner is a common technique when controlling a real robotic systems with physical constraints like maximum velocities [17], [45], [46].

1) Single-Feature Gradient EKF Gradients: From the error covariance update (5), only the correction term that includes the Kalman gain, $\mathbf{G}_{k+1}$, contains the state estimate. In fact, we are only interested in the linearized measurement matrix, $\mathbf{H}_{k+1}$, evaluated with the state estimate at time-step $k+1$

$$\mathbf{H}_{k+1} = \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k+1}^-} = \mathbf{K}\begin{bmatrix} \frac{1}{\hat{x}_{k+1,z}^-} & 0 & -\frac{\hat{x}_{k+1,x}^-}{\left(\hat{x}_{k+1,z}^-\right)^2} \\ 0 & \frac{1}{\hat{x}_{k+1,z}^-} & -\frac{\hat{x}_{k+1,y}^-}{\left(\hat{x}_{k+1,z}^-\right)^2} \\ 0 & 0 & 0 \end{bmatrix}. \tag{28}$$

Therefore, we are interested in evaluating the partial derivative of $\mathrm{Tr}(\Sigma_{k+1})$ from (6). Recall that Σ_{k+1}^- is a deterministic function of the current covariance, Σ_k. The rest of the gradient takes the general form shown in Section III-A. Finally, the partial derivatives of $\mathbf{H}_{k+1}$ with respect to each estimated state are given by

$$\frac{\partial \mathbf{H}_{k+1}}{\partial \hat{x}_x} = \mathbf{K}\begin{bmatrix} 0 & 0 & -\frac{1}{\left(\hat{x}_z^-\right)^2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{H}_{k+1}}{\partial \hat{x}_y} = \mathbf{K}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\left(\hat{x}_z^-\right)^2} \\ 0 & 0 & 0 \end{bmatrix} \tag{29}$$

and

$$\frac{\partial \mathbf{H}_{k+1}}{\partial \hat{x}_z} = \mathbf{K}\begin{bmatrix} -\frac{1}{\left(\hat{x}_z^-\right)^2} & 0 & 2\frac{\hat{x}_x^-}{\left(\hat{x}_z^-\right)^3} \\ 0 & -\frac{1}{\left(\hat{x}_z^-\right)^2} & 2\frac{\hat{x}_y^-}{\left(\hat{x}_z^-\right)^3} \\ 0 & 0 & 0 \end{bmatrix}.$$

The final instantaneous control is given by (26). It consists of a matrix multiplication that includes the closed-form partial derivatives given in (6) and (29). The final control only requires the previous state estimate update, $\hat{\mathbf{x}}_k$, the current state estimate prediction, $\hat{\mathbf{x}}_k^-$, and the current error covariance, Σ_k.

2) Single-Feature UKF Gradients: The expression for the gradient descent using the UKF is calculated starting with (14). The gradient for the UKF takes the general form shown in Section III. One major difference is that the prediction step may be completed using the standard Kalman filter prediction since the system dynamics are linear. The sigma vectors are constructed after the prediction in order to effectively use the UKF on the nonlinear portion of the system. In this 3-D vision example, the specific partial derivatives for the update are

given by

$$\frac{\partial h(\chi_{k+1}^{[i]-})}{\partial \chi_{k+1,x}^{[i]-}} = \mathbf{K}\begin{bmatrix} \frac{1}{\chi_{k+1,z}^{[i]-}} \\ 0 \\ 0 \end{bmatrix}$$
$$\frac{\partial h(\chi_{k+1}^{[i]-})}{\partial \chi_{k+1,y}^{[i]-}} = \mathbf{K}\begin{bmatrix} 0 \\ \frac{1}{\chi_{k+1,z}^{[i]-}} \\ 0 \end{bmatrix} \tag{30}$$

and

$$\frac{\partial h(\chi_{k+1}^{[i]-})}{\partial \chi_{k+1,z}^{[i]-}} = \mathbf{K}\begin{bmatrix} \frac{-\chi_{k+1,x}^{[i]-}}{\left(\chi_{k+1,z}^{[i]-}\right)^2} \\ \frac{-\chi_{k+1,y}^{[i]-}}{\left(\chi_{k+1,z}^{[i]-}\right)^2} \\ 0 \end{bmatrix}.$$

Equation (16) evaluates to the columns of $(e^{-\Omega_k \Delta t} - \mathbf{I}_n)$ in this case.

The final instantaneous control is again given by (26) and now consists of a matrix multiplication that includes the closed-form partial derivatives given in (14) and (30). The final control also depends on only the previous state estimate update, $\hat{\mathbf{x}}_k$, the current state estimate prediction, $\hat{\mathbf{x}}_k^-$, and the current error covariance, Σ_k.

3) Multiple-Feature Gradients : When dealing with M features, we denote the state vector as $\mathbf{x} = [\mathbf{x}_{[1]}, \ldots, \mathbf{x}_{[l]}, \ldots, \mathbf{x}_{[M]}]^T \in \mathbb{R}^n$, where $\mathbf{x}$ is of size $n = 3M$ and $\mathbf{x}_{[l]} = [x_{[l]_x}, x_{[l]_y}, x_{[l]_z}]^T \in \mathbb{R}^3$ is the lth feature block. Similarly, the measurement vector is now $\mathbf{y} = [\mathbf{y}_{[1]}, \ldots, \mathbf{y}_{[l]}, \ldots, \mathbf{y}_{[M]}]^T \in \mathbb{R}^m$, where $\mathbf{y}$ is of size $m = 2M$ and $\mathbf{y}_{[l]} = [y_{[l]_x}, y_{[l]_y}]^T \in \mathbb{R}^2$.

The motion and measurement models are written more generally as

$$\begin{aligned} \mathbf{x}_{k+1} &= (\mathbf{I}_M \otimes e^{-\Omega_k \Delta t})\mathbf{x}_k - (\mathbf{1}_M \otimes \mathbf{u}_k \Delta t) + \mathbf{v} \\ \mathbf{y}_k &= (\mathbf{I}_M \otimes \frac{1}{x_z}\mathbf{K})\mathbf{x}_k + \mathbf{w}. \end{aligned} \tag{31}$$

In this fashion, all estimation predictions and updates will remain the same as the single feature variant. Similarly, the active estimation gradients for the EKF and UKF are computed for each lth subsystem with (29) and (30), respectively.

The active sensing controller from (26) will take the following form:

$$(\mathbf{1}_M \otimes \mathbf{u}_k) = (\mathbf{I}_M \otimes e^{-\Omega \Delta t})\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k + \boldsymbol{\nabla} \tag{32}$$

where $\boldsymbol{\nabla}$ now has dimensions of $3M \times 1$. Note that each 3-D subsystem will contain the same control inputs since we are only moving one camera, therefore this is an overdetermined system. Equation (32) may then be conveniently rearranged as

$$\begin{bmatrix} \mathbf{I}_3 & -(e^{-\Omega \Delta t} - \mathbf{I}_3)\hat{\mathbf{x}}_{k,[1]} - \boldsymbol{\nabla}_{[1]} \\ \vdots & \vdots \\ \mathbf{I}_3 & -(e^{-\Omega \Delta t} - \mathbf{I}_3)\hat{\mathbf{x}}_{k,[l]} - \boldsymbol{\nabla}_{[l]} \\ \vdots & \vdots \\ \mathbf{I}_3 & -(e^{-\Omega \Delta t} - \mathbf{I}_3)\hat{\mathbf{x}}_{k,[M]} - \boldsymbol{\nabla}_{[M]} \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ 1 \end{bmatrix} = \mathbf{0}_{3n \times 1} \tag{33}$$

which takes the form $\mathbf{A}[\mathbf{u}_k^T, 1]^T = \mathbf{0}$, where $\mathbf{A} \in \mathbb{R}^{3n \times 4}$ and $[\mathbf{u}_k^T, 1]^T \in \mathbb{R}^{4 \times 1}$. We are ultimately interested in finding a $\mathbf{u}_k$ such that $[\mathbf{u}_k^T, 1]^T$ lies in the nullspace of $\mathbf{A}$. Suppose for a moment that we constrain the vector $[\mathbf{u}_k^T, 1]^T$ to a unit vector, i.e., $\|[\mathbf{u}_k^T, 1]^T\|_2 = 1$, then we could compute the linear least squares estimate of $\mathbf{u}_k$ via the singular value decomposition (SVD) of $\mathbf{A}$. The final solution for $\mathbf{u}_k$ can then be recovered by scaling the solution from the SVD such that the last component equals 1, as is desired from (33). Unfortunately, the SVD is expensive to compute when n is large. Furthermore, a good choice for $\mathbf{u}_k$ should intuitively be one that yields the minimum error between $\mathbf{u}_k$ and each lth term in matrix $\mathbf{A}$. Therefore, we present the following theorem to solve for the control $\mathbf{u}_k$.

Theorem 1 (Analytical Solution for Velocity): Suppose a matrix $\mathbf{A}$ in the equation $\mathbf{A}[\mathbf{u}_k^T, 1]^T = \mathbf{0}$ takes the form as shown in (33). Then, the least squares estimate $\mathbf{u}_k$ is

$$\mathbf{u}_k^* = \frac{1}{M}\sum_{l=1}^{M} \boldsymbol{\psi}_{[l]} \tag{34}$$

where $\boldsymbol{\psi}_{[l]} = (e^{-\Omega_k \Delta t} - \mathbf{I}_3)\hat{\mathbf{x}}_{k,[l]} + \boldsymbol{\nabla}_{[l]}$.

Proof: Consider the least squares estimation problem

$$\mathbf{u}_k^* = \arg\min_{\mathbf{u}_k} \left\| \mathbf{0} - \mathbf{A}[\mathbf{u}_k^T, 1]^T \right\|_2^2. \tag{35}$$

The objective in this minimization may also be written as

$$\begin{aligned} &\left\| \mathbf{A}[\mathbf{u}_k^T, 1]^T \right\|_2^2 \\ &\quad = \sum_{l=1}^{M} \|\mathbf{I}_3 \mathbf{u}_k - \boldsymbol{\psi}_{[l]}\|_2^2 \\ &\quad = \sum_{l=1}^{M} (\mathbf{u}_k - \boldsymbol{\psi}_{[l]})^T (\mathbf{u}_k - \boldsymbol{\psi}_{[l]}) \\ &\quad = M\mathbf{u}_k^T \mathbf{u}_k - 2\mathbf{u}_k^T \left(\sum_{l=1}^{M} \boldsymbol{\psi}_{[l]} \right) + \sum_{l=1}^{M} \boldsymbol{\psi}_{[l]}^T \boldsymbol{\psi}_{[l]}. \end{aligned} \tag{36}$$

The objective is quadratic in $\mathbf{u}_k$, thus we may set the derivative equal to zero and solve for minimizing $\mathbf{u}_k$. Doing so results in the equation $2M\mathbf{u}_k = 2\sum_{l=1}^{M} \boldsymbol{\psi}_{[l]}$ and thus yields the solution in (34). □

Equation (34) in Theorem 1 gives a fast closed-form expression for the velocity input without using a computationally expensive SVD. Note that the components of $\boldsymbol{\nabla}$ are computed in the same manner as the previous section. The only difference being the partial derivatives of (29) which will become large, sparse, block-diagonal $n \times n$ matrices. Only the indices corresponding to the states of feature l will be populated with the matrices in (29).

V. Results

In this section, we demonstrate the active EKF and UKF on the vision-based control application presented in Section IV. First, we introduce the various control methods to which our proposed vision-based gradient controller is compared, including a random walk strategy, a "NBV" greedy action selection, a greedy receding horizon control, and the active

SfM controller presented in [17] and [18]. We test the active filter schemes in simulations and hardware experiments with multiple image features.

A. Control Methods for Comparison

1) Random Walk: The *random walk* control applies a randomly selected unit vector $\mathbf{u}_{\text{random}} \in \mathbb{U}$ for a random amount of time-steps $t_{\text{rand}} \sim \mathcal{U}[1, 10]$. This random walk method allows for the camera to move a significant distance in any direction, making it a more competitive control strategy for estimation.

2) Greedy: The *greedy control* is computed by solving the following minimization:

$$\mathbf{u}_{\text{greedy}} = \underset{\mathbf{u}_k \in \mathbb{U}_d}{\arg\min}\ \text{Tr}(\Sigma_{k+1}) \tag{37}$$

where $\mathbb{U}_d$ is a discretized action set of dimension d. The size of the action set d—the resolution of the control discretization—is determined offline such that the solution time of (37) is slightly higher than the solution of the active gradients (6) and (14) giving this method more than fair conditions for success. In practice, we use an action space of size 200 and 20 when solving (37) for the EKF and UKF, respectively. The action space for the greedy selection UKF is smaller than the EKF because Σ_{k+1} is, in general, more computationally intensive to compute for the UKF, allowing far less potential actions to be explored. If we infinitely discretized the action space, this control should match our gradient descent method.

3) Receding Horizon: The *greedy receding horizon* control applies the solution of

$$\mathbf{u}_{\text{rec hor}} = \underset{\mathbf{u}_k \in \mathbb{U}_d}{\arg\min} \sum_{i=k+1}^{k+N} \gamma^{(i-k)} \text{Tr}(\Sigma_i) \tag{38}$$

to the camera motion, where the actions space is similarly discretized to the greedy selection case. The reward discount $\gamma \in (0, 1]$ helps decay the cost exponentially with time to favor more immediate rewards. We set $\gamma = 0.9$ in the simulations. In a similar fashion to the greedy control, the following sizes are chosen to keep the per-loop computation approximately one order of magnitude larger than the gradient controller: the size of the action space is $d = 12$ for both estimators and the horizon is $N = 3$ and $N = 2$ for the EKF and UKF, respectively.

4) Active SfM: We compare our method to the active sensing controller from [17] and [18], which tunes the transient response of a nonlinear observer in real time in order to decrease the estimation error. This work was directly applied to our 3-D reconstruction problem with the 6DOF camera (instead of the quadrotor) and the pinhole camera model. We have implemented this nonlinear observer in our quadrotor simulation and compute their active SfM control input as $\mathbf{u}_k^{\text{des}}$ as the desired body-frame velocity to be tracked by our quadrotor. We simultaneously compute our estimator (EKF or UKF) to compare the controller's success in minimizing our objective function based on the covariance since the nonlinear observer has no equivalent notion of uncertainty.

5) Gradient: The *gradient* controller utilizes the method presented in Section IV.

B. Simulations

We utilize a simulation environment in MATLAB to demonstrate the reconstruction of a complicated outdoor scene with a high-resolution camera on an aerial quadrotor. The *object* in the environment consists of M 3-D points (see Fig. 1) that are generated according to a uniform distribution with support defined by the rectangular volume of $1.5 \times 1.5 \times 3\,\text{m}^3$. We also define one of the points to be located at the origin of the world coordinate system for comparison purposes. Although our algorithm can handle objects with clear structure, we choose to validate on random points to ensure functionality with a generic 3-D scene. The 1000×1000 virtual camera image is captured using the pinhole camera model, where the 3-D points are projected onto the 2-D virtual image plane using (23) and the following camera calibration matrix:

$$\mathbf{K} = \begin{bmatrix} 500 & 0 & 500 \\ 0 & 500 & 500 \end{bmatrix}. \tag{39}$$

We model a quadrotor with a $0.25-\text{m}$ arm length, weighing approximately 0.9 kg with the following moment of inertia matrix:

$$\mathbf{J} = \begin{bmatrix} 0.0048 & 0.0000 & 0.0001 \\ 0.0000 & 0.0048 & 0.0000 \\ 0.0001 & 0.0000 & 0.0083 \end{bmatrix} \frac{\text{kg} \cdot \text{m}}{\text{s}^2}. \tag{40}$$

This is the model of the research quadrotor used in the Multi-Robot Systems Lab at Stanford University and in the experiments of this paper. The quadrotor control is computed according to (22) at 200 Hz while the desired translational velocity control input is computed at 5 Hz. Simulated noise is added to the dynamics and image measurements according to $\mathcal{N}(\mathbf{0}, 0.03\,\mathbf{I}_{3n}\,(\text{m/s})^2)$ and $\mathcal{N}(\mathbf{0}, \mathbf{I}_{2n}\,\text{pixels}^2)$, respectively. We additionally add noise to the rotation of the quadrotor according to $\mathcal{N}(\mathbf{0}, 0.001\,\mathbf{I}_{3n}\,(\text{rad/s})^2)$. The filters and control are implemented as shown in Section IV with covariance matrices $\mathbf{Q} = 0.0001\,\mathbf{I}_{3n}\,(\text{m/s})^2$ and $\mathbf{R} = \mathbf{I}_{2n}\,\text{pixels}^2$ while the initial depth estimate is selected randomly according to a uniform distribution with support in the interval of $[1, 50]$ m. The other states, $\hat{x}_{0,x}$ and $\hat{x}_{0,y}$, are initialized given these values and the measurement equation in (23). The control gain in (27) is set to $v^{\text{des}} = 0.09\text{m/s}$. In general, higher gains will increase the convergence time of the filter, but we artificially limit it here to better observe the effects of different controllers. Finally, the small normalization gain is $\epsilon = 0.005$.

We compare all of the methods from Section V-A in this simulation with EKF and UKF separately. In each comparison trial, we compute a random quadrotor initial condition (position and linear velocity), a random object consisting of 25 world features, and a random filter initialization. The camera's position is initialized according to a uniform distribution with support defined by the rectangular volume $10 \times 10 \times 5\,\text{m}^3$, which is centered 10 m above the origin of the environment. One-hundred such trails are computed for all methods while tracking the 25 feature in the image frame. The results for the EKF and UKF are shown in Figs. 1 and 2, respectively.

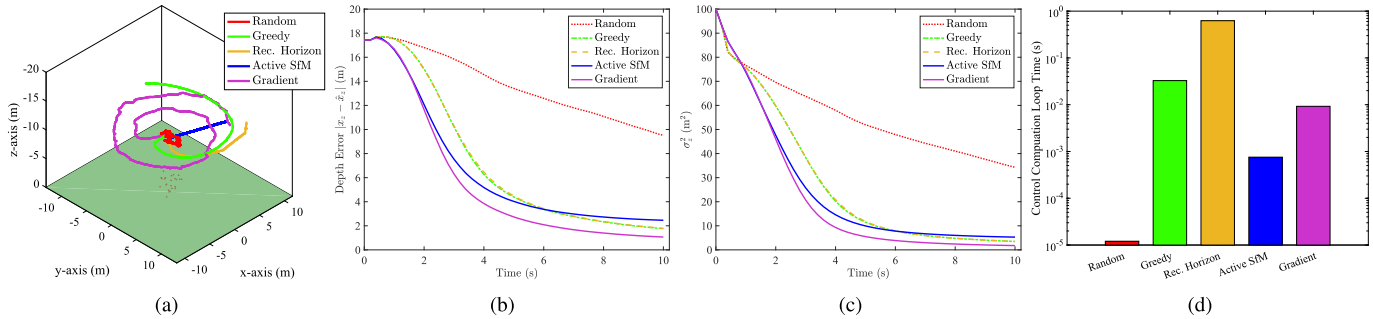


Fig. 1. Simulation results comparing random walk, greedy, receding horizon, active SfM [17], [18], and the active EKF methods over 100 random trials using 25 image features. (a) One example trajectory for each control strategy. (b) Mean of the estimated feature depths for each method. (c) Mean of the error covariances. Only the first 10 s of the simulation are shown for clarity. Our active gradient approach lowers the estimation error and uncertainty more quickly than all other methods. (d) Mean control computation loop time comparison is plotted in log-scale. Although the active SfM runs faster, our method runs at approximately 100 Hz while estimating the 3-D scene with less error.

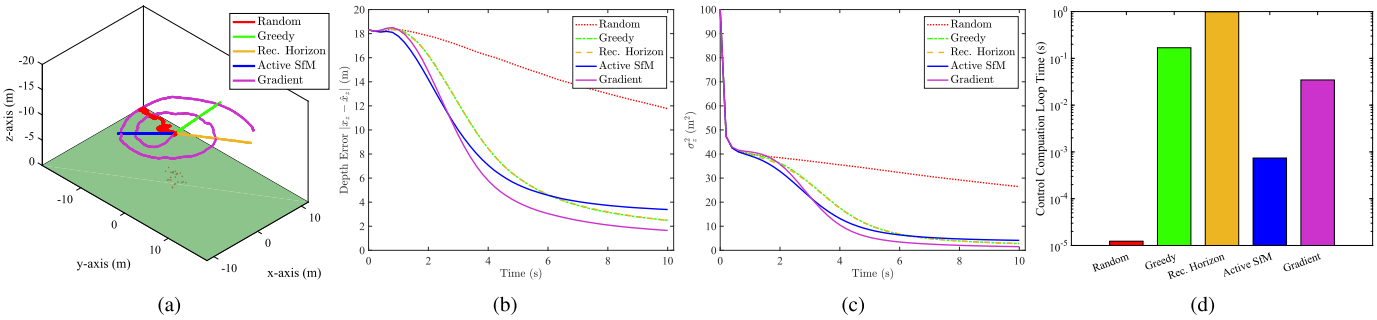
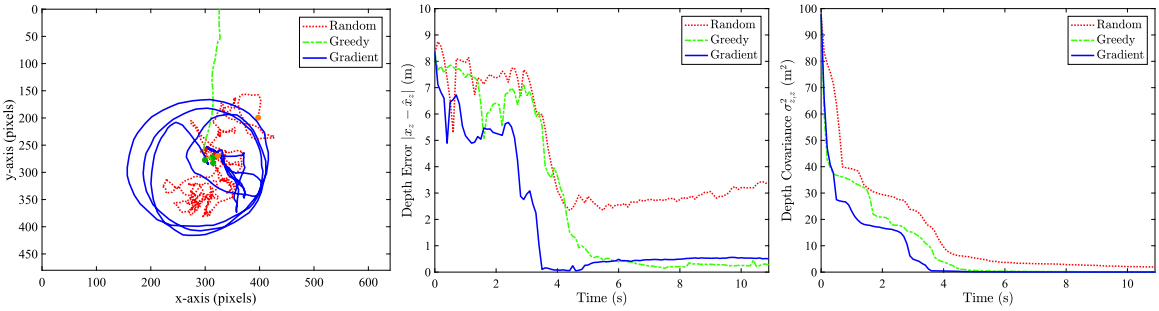


Fig. 2. Simulation results comparing the random walk, greedy, receding horizon, and active SfM [17], [18], and the active UKF methods over 100 random trials using 25 image features. (a) One example trajectory for each control strategy. (b) Mean of the estimated feature depths for each method. (c) Mean of the error covariances. Again, only the first 10 s of the simulation is shown for clarity. Our active gradient reaches the lowest estimation error and uncertainty at the end of the simulation. (d) Mean control computation loop time comparison is plotted in log-scale. Note here that the active SfM computation does not increase since it does not consider a Bayesian estimate of the scene. The active UKF is more computationally demanding than the active EKF but provides the lowest uncertainty of the unknown 3-D scene.

Figs. 1(a) and 2(a) illustrate the example quadrotor trajectories from each control for approximately 10 min of flight. Interestingly, an outward spiral emerges from the active 3-D reconstruction policies that consider the covariance matrix in the objective. Of the control methods, our gradient approach executes the *tightest* spiral motion which is advantageous for two reasons: 1) obtaining image measurements for longer will only continue to improve the estimation and 2) moving the images in a spiral about the outside of the image produces the most image feature motion—increased parallax—without losing sight of the features. In general, the greedy, receding horizon, and active SfM move the image features away from the center more aggressively which eventually leads to the loss of the image features from the field-of-view. The trajectories for both greedy methods with the UKF is a line instead of a spiral due to discretization since the action space is significantly more coarse in the receding horizon case to keep the problem tractable. The trajectories produced from the active SfM are always straight lines since the camera's attitude is coupled in the quadrotor dynamics. This is to say, we do not use a separate attitude controller as is used in [17] to keep the image features in the center of the image which enables the 3-D circular motion shown in their work.

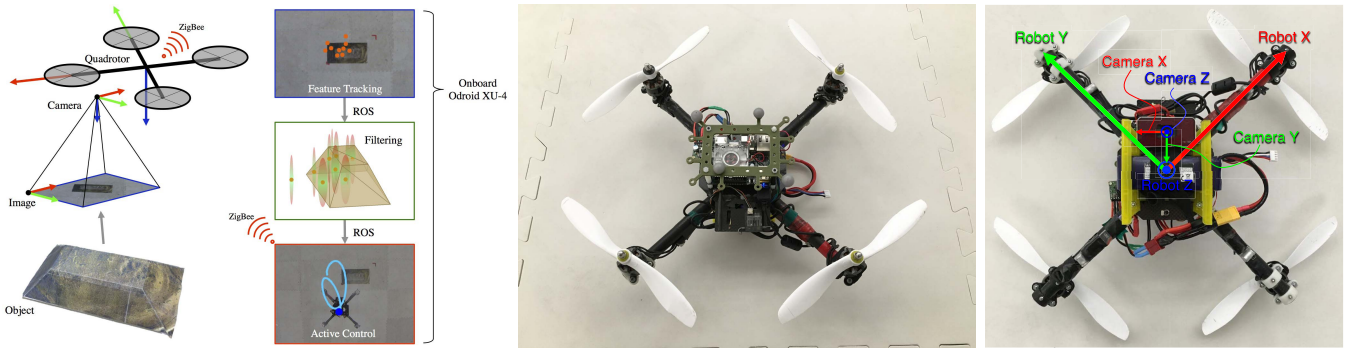
We compare the mean over the 100 trials of the average features' depth estimation error, $|x_z - \hat{\mathbf{x}}_z|$, in Figs. 1(b) and 2(b), and autocovariance corresponding to the depth estimate, $\sigma_{z,z}^2$, for both methods shown in Figs. 1(c) and 2(c). For both filters, any of the solutions outperforms a random walk. Our gradient-based active estimators outperform the greedy control action (single and multiple horizon) selection both in terms of depth estimation error reduction and error covariance reduction (as defined in our cost function). It is interesting to note that the one-step greedy performs nearly identically to the receding horizon control. We also attribute this to the coarse discretization of the action space and limited planning horizon.

The active SfM method has the most competitive performance in terms of estimation error and uncertainty. Our active gradient outperforms this method easily using the EKF. Using a UKF, the active SfM performs well initially but surprisingly is the second worst performing at the end of the simulation. For both filters, the active SfM actually produces the second poorest performance at the end of the 10 sec. We hypothesis that the difference in trajectories and estimator performance with the active SfM compared to our method may indicate a difference between using covariance matrix objectives versus strategies that consider the Fisher information matrix. Both metrics are concerned with lowering the uncertainty of the estimate, but our gradient approach appears to lower the uncertainty more quickly. Most importantly, while the filter performance is similar, the original active SfM method



(a) (b) (c)

Fig. 3. Quadrotor experiment results comparing the random walk, greedy, and our active EKF using 10 image features. (a) One example trajectory for each control strategy of the image features in the image plane. We note the same spiraling trajectory observed in the simulation study. (b) Mean of the estimated feature depths for each method. (c) Mean of the error covariances. Only the first 10 s of the simulation are shown for clarity as well. As anticipated, the active EKF lowers the uncertainty more quickly than the other two methods.



(a) (b) (c)

Fig. 4. Experimental setup including a KMel K500 quadrotor with onboard Odroid XU-4 and USB camera. (a) Experimental system diagram. (b) Quadrotor with onboard computer. (c) Coordinate frames.

estimates the unknown feature depth via nonlinear observer which does not reason about the uncertainty in a Bayesian sense as we do.

The estimation error reduction for our active EKF and active UKF is similar. In terms of only estimation performance, the active UKF is advantageous because the estimation error covariance of feature depth is 57.6% lower than with the active EKF at the end of the simulation, over all trials. The EKF is known to perform less accurately under poor initial conditions and we believe this is a significant advantage of the active UKF. The tradeoff for this accuracy is the computational increase required to compute the active UKF (26.6% slower than the active EKF). In either case, it is apparent that the active filters outperform the other control methods at the end of 10 s despite the UKF's initial poor estimate.

Finally, the mean computation time over all trials for both simulations is shown in Figs. 1(d) and 2(d) and clearly shows the receding horizon greedy control as the slowest while the active SfM is the quickest—excluding the random walk. This being said, our proposed controllers can both run at least as fast as a typical camera (active EKF: 109 Hz and active UKF: 29 Hz) so neither would be the bottle neck in an onboard system. The active UKF is slower due to the evaluation of the partials with respect to each particle—a task that could easily be parallelized in the future as well. However, recall that our method estimates a Gaussian distribution over the state while the active SfM does not. Hence, the active SfM's computation does not increase when switching filters. If this estimation is critical, the UKF's more accurate approximation may be worth the computational penalty. The action spaces and horizons for the greedy controllers are designed to give them the best chance possible in terms of computation time compared to the active sensing gradient controller. Despite these advantages, neither greedy approach outperforms our method.

C. Experiments

We also conduct experimental trials with a quadrotor and camera that compare the active EKF controller to the other control methods. This experiment is a proof of concept to show that we can, indeed, achieve similar sparse reconstructions in under 10 sec as in the simulations. We have eliminated the receding horizon controller from this study because it clearly performed worse in the simulations in terms of computation time and estimation quality. We also only focus our efforts on the active EKF, as we expect the active UKF will yielding lower error for a slight computational penalty.

The experimental platform is a KMel k500 quadrotor equipped with a Hardkernel Odroid XU-4 and downward facing USB camera (Fig. 4). The Odroid is a microcomputer with a 2- GHzSamsung mobile application processor and a Cortex-A7 ARM processor. It has 2 GB of RAM and 64 GB of embedded multimedia controller (eMMC) flash storage. It runs Ubuntu 14.04, using OpenCV 2.4.3, and Robot Operating System (ROS) indigo. The quadrotor is flying above a 3-D object on the floor with sufficient texture and color for feature tracking in the shape of a 3-D frustrum fitting into a volume of approximately $0.5 \times 0.2 \times 0.2\,\mathrm{m}^3$.

The entire control pipeline is illustrated in Fig. 4(a) and is implemented using ROS online and onboard the Odroid. Image acquisition occurs at approximately 20 Hz. The feature tracking node subscribes to the image feed and tracks OpenCV's Lucas–Kanade features. We run the EKF in a stand-alone node that subscribes to the motion capture system for the IMU data and to the feature tracker for measurements. Finally, the active control node subscribes to the filter and computes the gradients.

The low-level quadrotor control utilizes differential flatness [44] and is performed in MATLAB where an ROS node subscribes to the latest calculated control inputs. Feature tracking and state estimation occur at approximately 15 Hz. The quadrotor body-frame velocities are estimated for differentiating data from a motion capture system (Optitrack) captured at approximately 60 Hz with added Gaussian noise.

We compare the controllers using ten image features. The EKF uses covariance matrices, $\mathbf{Q} = 0.0001\mathbf{I}_{3M}\,(\mathrm{m/s})^2$ and $\mathbf{R} = 25\mathbf{I}_{2M}\,\mathrm{pixels}^2$ while the initial depth estimate is 3 m (compared to true camera height of approximately 2 m). As expected, the active EKF outperforms the random walk control and greedy strategies (see Fig. 3). Although noisier, we observe the similar trend where the gradient's estimation covariance seems to lower bound the estimation from the other strategies. Interestingly, we note that the estimation error increases slightly after nearly reducing to zero. This appears to be a function of the controller in (27) that performs the soft saturation. In contrast, the greedy approach simply keeps moving forward in a motion favored by the estimator. In practice, the robot could move on to other tasks at this time since it is confident in its estimation of the unknown structure.

VI. Conclusion

We have presented a computationally efficient, gradient-based control strategy that actively estimates the unknown depth of features in images using either an EKF or UKF. The controllers calculate the translational velocity for the robot using an explicit expression of the image feature position, its 3-D state estimate from the nonlinear filter, and its error covariance matrix. We have shown in both simulations and experiments that they perform better than a random, greedy, receding horizon, or active SfM approach, making this a suitable controller for quick, sparse 3-D reconstruction onboard a lightweight MAV. Future directions for this paper include extending the method to multiple collaborating quadrotors, incorporating a forward facing camera, and dealing explicitly with occlusions and more complex scene geometries. We also hope to integrate this system with other tasks based on the 3-D scene estimate, such as object classification or collision avoidance.

Appendix

A. Active Extended Kalman Filter Derivation

The derivation of Proposition 1 is given in this section. In the derivation, we make the following substitution for simplicity $\mathbf{M}_{k+1} = \mathbf{H}_{k+1}\Sigma_{k+1}^{-}\mathbf{H}_{k+1}^{T} + \mathbf{R}$. The jth component of the gradient is

$$\begin{aligned}\nabla_j &= \frac{\partial \mathrm{Tr}(\Sigma_{k+1})}{\partial \hat{x}_j} = \mathrm{Tr}\left(\frac{\partial \Sigma_{k+1}}{\partial \hat{x}_j}\right)\\ &= \mathrm{Tr}\left(\frac{\partial}{\partial \hat{x}_j}\left(\Sigma_{k+1}^{-} - \Sigma_{k+1}^{-}\mathbf{H}_{k+1}^{T}\mathbf{M}_{k+1}^{-1}\mathbf{H}_{k+1}\Sigma_{k+1}^{-}\right)\right).\end{aligned}$$

Developing yields

$$\begin{aligned}\nabla_j = \mathrm{Tr}\Bigg(&\frac{\partial \Sigma_{k+1}^{-}}{\partial \hat{x}_j} - \frac{\partial \Sigma_{k+1}^{-}}{\partial \hat{x}_j}\mathbf{H}_{k+1}^{T}\mathbf{M}_{k+1}^{-1}\mathbf{H}_{k+1}\Sigma_{k+1}^{-}\\ &-\Sigma_{k+1}^{-}\left(\frac{\partial \mathbf{H}_{k+1}}{\partial \hat{x}_j}\right)^{T}\mathbf{M}_{k+1}^{-1}\mathbf{H}_{k+1}\Sigma_{k+1}^{-}\\ &-\Sigma_{k+1}^{-}\mathbf{H}_{k+1}^{T}\left(-\mathbf{M}_{k+1}^{-1}\frac{\partial \mathbf{M}_{k+1}}{\partial \hat{x}_j}\mathbf{M}_{k+1}^{-1}\right)\mathbf{H}_{k+1}\Sigma_{k+1}^{-}\\ &-\Sigma_{k+1}^{-}\mathbf{H}_{k+1}^{T}\mathbf{M}_{k+1}^{-1}\frac{\partial \mathbf{H}_{k+1}}{\partial \hat{x}_j}\Sigma_{k+1}^{-}\\ &-\Sigma_{k+1}^{-}\mathbf{H}_{k+1}^{T}\mathbf{M}_{k+1}^{-1}\mathbf{H}_{k+1}\frac{\partial \Sigma_{k+1}^{-}}{\partial \hat{x}_j}\Bigg).\end{aligned}$$

Since the trace is not affected by the transpose operator, the previous expression can be further simplified

$$\begin{aligned}\nabla_j = \mathrm{Tr}\Bigg(&\frac{\partial \Sigma_{k+1}^{-}}{\partial \hat{x}_j} - 2\Sigma_{k+1}^{-}\mathbf{H}_{k+1}^{T}\mathbf{M}_{k+1}^{-1}\mathbf{H}_{k+1}\frac{\partial \Sigma_{k+1}^{-}}{\partial \hat{x}_j}\\ &-2\Sigma_{k+1}^{-}\mathbf{H}_{k+1}^{T}\mathbf{M}_{k+1}^{-1}\frac{\partial \mathbf{H}_{k+1}}{\partial \hat{x}_j}\Sigma_{k+1}^{-}\\ &+\Sigma_{k+1}^{-}\mathbf{H}_{k+1}^{T}\mathbf{M}_{k+1}^{-1}\frac{\partial \mathbf{M}_{k+1}}{\partial \hat{x}_j}\mathbf{M}_{k+1}^{-1}\mathbf{H}_{k+1}\Sigma_{k+1}^{-}\Bigg)\\ = \mathrm{Tr}\Bigg(&\frac{\partial \Sigma_{k+1}^{-}}{\partial \hat{x}_j} - 2\mathbf{G}_{k+1}\mathbf{H}_{k+1}\frac{\partial \Sigma_{k+1}^{-}}{\partial \hat{x}_j}\\ &-2\mathbf{G}_{k+1}\frac{\partial \mathbf{H}_{k+1}}{\partial \hat{x}_j}\Sigma_{k+1}^{-} + \mathbf{G}_{k+1}\frac{\partial \mathbf{M}_{k+1}}{\partial \hat{x}_j}\mathbf{G}_{k+1}^{T}\Bigg).\end{aligned} \tag{41}$$

The partial of $\mathbf{M}_{k+1}$ is

$$\begin{aligned}\frac{\partial \mathbf{M}_{k+1}}{\partial \hat{x}_j} = &\frac{\partial \mathbf{H}_{k+1}}{\partial \hat{x}_j}\Sigma_{k+1}^{-}\mathbf{H}_{k+1}^{T} + \mathbf{H}_{k+1}\frac{\partial \Sigma_{k+1}^{-}}{\partial \hat{x}_j}\mathbf{H}_{k+1}^{T}\\ &+\mathbf{H}_{k+1}\Sigma_{k+1}^{-}\frac{\partial \mathbf{H}_{k+1}}{\partial \hat{x}_j}^{T}\end{aligned} \tag{42}$$

and the partial derivative of Σ_{k+1}^{-} is

$$\frac{\partial \Sigma_{k+1}^{-}}{\partial \hat{x}_j} = \frac{\partial \mathbf{F}_{k+1}}{\partial \hat{x}_j}\Sigma_k\mathbf{F}_{k+1}^{T} + \mathbf{F}_{k+1}\Sigma_k\frac{\partial \mathbf{F}_{k+1}}{\partial \hat{x}_j}^{T} \tag{43}$$

because Σ_k is not a function of $\hat{\mathbf{x}}_k$.

Finally, replacing (42) and (43) into (41) and using again the invariance of the trace to the transpose operator, (6) is obtained.

B. Active Unscented Kalman Filter Derivation

The derivation of proposition 2 is given in this section. While inspecting the cost function, $\mathrm{Tr}(\Sigma_{k+1})$, we remove the

time index $k+1$ on all variables for visual clarity. The jth component of the gradient is

$$\begin{aligned}\nabla_j &= \frac{\partial \mathrm{Tr}(\Sigma)}{\partial \hat{x}_j} = \mathrm{Tr}\left(\frac{\partial \Sigma}{\partial \hat{x}_j}\right) \\ &= \mathrm{Tr}\left(\frac{\partial \Sigma^-}{\partial \hat{x}_j} - \frac{\partial}{\partial \hat{x}_j}\left(\Sigma_{\mathbf{x},\mathbf{y}}\Sigma_{\mathbf{y},\mathbf{y}}^{-1}(\Sigma_{\mathbf{x},\mathbf{y}})^T\right)\right) \\ &= \mathrm{Tr}\left(\frac{\partial \Sigma^-}{\partial \hat{x}_j} - \left(\frac{\partial}{\partial \hat{x}_j}\Sigma_{\mathbf{x},\mathbf{y}}\right)\Sigma_{\mathbf{y},\mathbf{y}}^{-1}(\Sigma_{\mathbf{x},\mathbf{y}})^T \right. \\ &\qquad - \Sigma_{\mathbf{x},\mathbf{y}}\left(-\Sigma_{\mathbf{y},\mathbf{y}}^{-1}\left(\frac{\partial}{\partial \hat{x}_j}\Sigma_{\mathbf{y},\mathbf{y}}\right)\Sigma_{\mathbf{y},\mathbf{y}}^{-1}\right)(\Sigma_{\mathbf{x},\mathbf{y}})^T \\ &\qquad \left. - \Sigma_{\mathbf{x},\mathbf{y}}\Sigma_{\mathbf{y},\mathbf{y}}^{-1}\left(\frac{\partial}{\partial \hat{x}_j}\Sigma_{\mathbf{x},\mathbf{y}}\right)^T\right) \\ &= \mathrm{Tr}\left(\frac{\partial \Sigma^-}{\partial \hat{x}_j} - 2\left(\frac{\partial}{\partial \hat{x}_j}\Sigma_{\mathbf{x},\mathbf{y}}\right)\Sigma_{\mathbf{y},\mathbf{y}}^{-1}(\Sigma_{\mathbf{x},\mathbf{y}})^T \right. \\ &\qquad \left. - \Sigma_{\mathbf{x},\mathbf{y}}\left(-\Sigma_{\mathbf{y},\mathbf{y}}^{-1}\left(\frac{\partial}{\partial \hat{x}_j}\Sigma_{\mathbf{y},\mathbf{y}}\right)\Sigma_{\mathbf{y},\mathbf{y}}^{-1}\right)(\Sigma_{\mathbf{x},\mathbf{y}})^T\right). \end{aligned} \tag{44}$$

For the gradient calculation of Σ_{k+1}, we recall that the sigma points are first defined by the previous state update at time k following (11). The sigma points are then propagated through the nonlinear dynamics and the state prediction is estimated by the weighted sum of these new sigma vectors (12). The first component of the gradient is

$$\begin{aligned}\frac{\partial \Sigma^-}{\partial \hat{x}_j} &= \frac{\partial}{\partial \hat{x}_j}([\mathbf{1}_N \otimes \hat{\mathbf{x}}^- - \boldsymbol{\chi}^-]\mathbf{W}[\mathbf{1}_N \otimes \hat{\mathbf{x}}^- - \boldsymbol{\chi}^-]^T) \\ &= \left[\mathbf{1}_N \otimes \left(\frac{\partial \boldsymbol{\chi}^-}{\partial \hat{x}_j}\boldsymbol{w}\right) - \frac{\partial \boldsymbol{\chi}^-}{\partial \hat{x}_j}\right]\mathbf{W}[\mathbf{1}_N \otimes \hat{\mathbf{x}}^- - \boldsymbol{\chi}^-]^T \\ &\quad + [\mathbf{1}_N \otimes \hat{\mathbf{x}}^- - \boldsymbol{\chi}^-]\mathbf{W}!\left[\mathbf{1}_N \otimes \left(\frac{\partial \boldsymbol{\chi}^-}{\partial \hat{x}_j}\boldsymbol{w}\right) - \frac{\partial \boldsymbol{\chi}^-}{\partial \hat{x}_j}\right]^T \end{aligned} \tag{45}$$

where the partials of the sigma points are given by

$$\frac{\partial \boldsymbol{\chi}^-}{\partial \hat{x}_j} = \left[\frac{\partial f\left(\chi_k^{[0]}, \mathbf{u}_k\right)}{\partial \hat{x}_j}, \ldots, \frac{\partial f\left(\chi_k^{[2n]}, \mathbf{u}_k\right)}{\partial \hat{x}_j}\right] \tag{46}$$

and

$$\frac{\partial f\left(\chi_k^{[i]}, \mathbf{u}_k\right)}{\partial \hat{x}_j} = \frac{\partial f\left(\chi_k^{[i]}\right)}{\partial \chi_{k,j}^{[i]}}\frac{\partial \chi_{k,j}^{[i]}}{\partial \hat{x}_j}. \tag{47}$$

Note that (45) will equal zero if the motion model is linear since $[\mathbf{1}_N \otimes \partial \boldsymbol{\chi}^-/\partial \hat{x}_j \boldsymbol{w} - \partial \boldsymbol{\chi}^-/\partial \hat{x}_j] = \mathbf{0}$ due to the definition of the UKF weights.

The partial derivative of the cross covariance is

$$\begin{aligned}\frac{\partial \Sigma_{\mathbf{xy}}}{\partial \hat{x}_j} &= \frac{\partial}{\partial \hat{x}_j}([\mathbf{1}_N \otimes \hat{\mathbf{x}}^- - \boldsymbol{\chi}^-]\mathbf{W}[\mathbf{1}_N \otimes \hat{\mathbf{y}}^- - \boldsymbol{y}^-)]^T) \\ &= \left[\mathbf{1}_N \otimes \left(\frac{\partial \boldsymbol{\chi}^-}{\partial \hat{x}_j}\boldsymbol{w}\right) - \frac{\partial \boldsymbol{\chi}^-}{\partial \hat{x}_j}\right]\mathbf{W}[\mathbf{1}_N \otimes \hat{\mathbf{y}}^- - \boldsymbol{y}^-]^T \\ &\quad + [\mathbf{1}_N \otimes \hat{\mathbf{x}}^- - \boldsymbol{\chi}^-]\mathbf{W}\left[\mathbf{1}_N \otimes \left(\frac{\partial \boldsymbol{y}^-}{\partial \hat{x}_j}\boldsymbol{w}\right) - \frac{\partial \boldsymbol{y}^-}{\partial \hat{x}_j}\right]^T \end{aligned} \tag{48}$$

and the partial derivative of the measurement auto covariance is

$$\begin{aligned}\frac{\partial \Sigma_{\mathbf{yy}}}{\partial \hat{x}_j} &= \frac{\partial}{\partial \hat{x}_j}([\mathbf{1}_N \otimes \hat{\mathbf{y}}^- - \boldsymbol{y}^-]\mathbf{W}_c[\mathbf{1}_N \otimes \hat{\mathbf{y}}^- - \boldsymbol{y}^-]^T + \mathbf{R}) \\ &= \left[\mathbf{1}_N \otimes \left(\frac{\partial \boldsymbol{y}^-}{\partial \hat{x}_j}\boldsymbol{w}\right) - \frac{\partial \boldsymbol{y}^-}{\partial \hat{x}_j}\right]\mathbf{W}[\mathbf{1}_N \otimes \hat{\mathbf{y}}^- - \boldsymbol{y}^-)]^T \\ &\quad + [\mathbf{1}_N \otimes \hat{\mathbf{y}}^- - \boldsymbol{y}^-]\mathbf{W}\left[\mathbf{1}_N \otimes \left(\frac{\partial \boldsymbol{y}^-}{\partial \hat{x}_j}\boldsymbol{w}\right) - \frac{\partial \boldsymbol{y}^-}{\partial \hat{x}_j}\right]^T. \end{aligned} \tag{49}$$

The partials for the sigma vectors are now given by

$$\frac{\partial \boldsymbol{y}_{k+1}^-}{\partial \hat{x}_j} = \left[\frac{\partial h\left(\chi_{k+1}^{[0]-}\right)}{\partial \hat{x}_j}, \ldots, \frac{\partial h\left(\chi_{k+1}^{[2n]-}\right)}{\partial \hat{x}_j}\right] \tag{50}$$

and

$$\frac{\partial h\left(\chi_{k+1}^{[i]-}\right)}{\partial \hat{x}_j} = \frac{\partial h\left(\chi_{k+1}^{[i]-}\right)}{\partial \chi_{k+1,j}^{[i]-}}\frac{\partial \chi_{k+1,j}^{[i]-}}{\partial \hat{x}_j} \tag{51}$$

or equivalently

$$\frac{\partial h\left(\chi_{k+1}^{[i]-}\right)}{\partial \hat{x}_j} = \frac{\partial h\left(\chi_{k+1}^{[i]-}\right)}{\partial \chi_{k+1,j}^{[i]-}}\frac{\partial f\left(\chi_k^{[i]}, \mathbf{u}_k\right)}{\partial \hat{x}_j}. \tag{52}$$

REFERENCES

[1] H. Ding *et al.*, "A multi-resolution approach for discovery and 3-D modeling of archaeological sites using satellite imagery and a UAV-borne camera," in *Proc. Amer. Control Conf. (ACC)*, 2016, pp. 1359–1365.

[2] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, vol. 26. New York, NY, USA: Springer-Verlag, 2012.

[3] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in *Springer Handbook of Robotics*. Berlin, Germany: Springer-Verlag, 2008, pp. 871–889.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May/Jun. 2014, pp. 15–22.

[5] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May/Jun. 2014, pp. 2609–2616.

[6] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2003, p. 1403.

[7] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[8] P. Whaite and F. P. Ferrie, "Autonomous exploration: Driven by uncertainty," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 3, pp. 193–205, Mar. 1997.

[9] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-blackwellized particle filters," in *Proc. Robot., Sci. Syst.*, vol. 2. 2005, pp. 65–72.

[10] E. Dunn, J. Van Den Berg, and J.-M. Frahm, "Developing visual sensing strategies through next best view planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2009, pp. 4001–4008.

[11] V. A. Sujan and S. Dubowsky, "Visually guided cooperative robot actions based on information quality," *Auto. Robots*, vol. 19, no. 1, pp. 89–110, 2005.

[12] È. Marchand and F. Chaumette, "An autonomous active vision system for complete and accurate 3D scene reconstruction," *Int. J. Comput. Vis.*, vol. 32, no. 3, pp. 171–194, 1999.

[13] C. Hoppe *et al.*, "Photogrammetric camera network design for micro aerial vehicles," in *Proc. Comput. Vis. Winter Workshop (CVWW)*, vol. 8, 2012, pp. 1–8.

[14] C. Forster, M. Pizzoli, and D. Scaramuzza, "Appearance-based active, monocular, dense reconstruction for micro aerial vehicle," in *Proc. Robot., Sci. Syst (RSS)*, 2014.
[15] C. Mostegel, A. Wendel, and H. Bischof, "Active monocular localization: Towards autonomous monocular exploration for multirotor MAVs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May/Jun. 2014, pp. 3848–3855.
[16] A. J. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 3046–3052.
[17] R. Spica and P. R. Giordano, "A framework for active estimation: Application to structure from motion," in *Proc. 52nd IEEE Conf. Decis. Control (CDC)*, Dec. 2013, pp. 7647–7653.
[18] R. Spica, P. R. Giordano, and F. Chaumette, "Active structure from motion: Application to point, sphere, and cylinder," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1499–1513, Dec. 2014.
[19] P. R. Giordano, R. Spica, and F. Chaumette, "An active strategy for plane detection and estimation with a monocular camera," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 4755–4761.
[20] R. Spica, P. R. Giordano, and F. Chaumette, "Plane estimation by active vision from point features and image moments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 6003–6010.
[21] P. Salaris, R. Spica, P. R. Giordano, and P. Rives, "Online optimal active sensing control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May/Jun. 2017, pp. 672–678.
[22] M. Cognetti, P. Salaris, and P. Robuffo, "Optimal active sensing with process and measurement noise," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–8.
[23] M. Krstic and H.-H. Wang, "Design and stability analysis of extremum seeking feedback for general nonlinear systems," in *Proc. 36th IEEE Conf. Decis. Control (CDC)*, vol. 2, Dec. 1997, pp. 1743–1748.
[24] W. H. Moase, C. Manzie, and M. J. Brear, "Newton-like extremum-seeking for the control of thermoacoustic instability," *IEEE Trans. Autom. Control*, vol. 55, no. 9, pp. 2094–2105, Sep. 2010.
[25] Y. Zhang and N. Gans, "Simplex guided extremum seeking control for real-time optimization," in *Proc. IEEE Amer. Control Conf. (ACC)*, Jun. 2012, pp. 3377–3382.
[26] Y. Zhang, M. Rotea, and N. Gans, "Simplex guided extremum seeking control with convergence detection to improve global performance," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 4, pp. 1266–1278, Jul. 2016.
[27] Y. Zhang, J. Shen, and N. Gans, "Real-time optimization for eye-in-hand visual search," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 325–339, Apr. 2014.
[28] G. Olague and R. Mohr, "Optimal camera placement for accurate reconstruction," *Pattern Recognit.*, vol. 35, no. 4, pp. 927–944, 2002.
[29] C. K. Cowan and P. D. Kovesi, "Automatic sensor placement from vision task requirements," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-10, no. 3, pp. 407–416, May 1988.
[30] X. Yang, W.-A. Zhang, M. Z. Q. Chen, and L. Yu, "Hybrid sequential fusion estimation for asynchronous sensor network-based target tracking," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 2, pp. 669–676, Mar. 2017.
[31] K. Zhou and S. I. Roumeliotis, "Optimal motion strategies for range-only constrained multisensor target tracking," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1168–1185, Oct. 2008.
[32] M. E. Campbell and W. W. Whitacre, "Cooperative tracking using vision measurements on SeaScan UAVs," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 4, pp. 613–626, Jul. 2007.
[33] P. Vela, A. Betser, J. Malcolm, and A. Tannenbaum, "Vision-based range regulation of a leader-follower formation," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 2, pp. 442–448, Mar. 2009.
[34] F. Morbidi and G. L. Mariottini, "Active target tracking and cooperative localization for teams of aerial vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 5, pp. 1694–1707, Sep. 2013.
[35] H. Wei, W. Lu, P. Zhu, G. Huang, J. Leonard, and S. Ferrari, "Optimized visibility motion planning for target tracking and localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2014, pp. 76–82.
[36] M. J. Kochenderfer, *Decision Making Under Uncertainty: Theory Application*. Cambridge, MA, USA: MIT Press, 2015.
[37] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. IEEE Adapt. Syst. Signal Process., Commun., Control Symp. (AS-SPCC)*, Sep. 2000, pp. 153–158.
[38] T. H. Chung, J. W. Burdick, and R. M. Murray, "A decentralized motion coordination strategy for dynamic target tracking," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2006, pp. 2416–2422.
[39] V. Grabe, H. H. Bülthoff, and P. R. Giordano, "On-board velocity estimation and closed-loop control of a quadrotor uav based on optical flow," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2012, pp. 491–497.
[40] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 664–674, 2012.
[41] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 2520–2525.
[42] L. Kneip, A. Martinelli, S. Weiss, D. Scaramuzza, and R. Siegwart, "Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 4546–4553.
[43] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual–inertial odometry," *Int. J. Robot. Res.*, vol. 32, no. 6, pp. 690–711, May 2013.
[44] D. Zhou and M. Schwager, "Vector field following for quadrotors using differential flatness," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 6567–6572.
[45] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," in *Proc. Int. Symp. Robot. Res. (ISRR)*. Cham, Switzerland: Springer, 2011, pp. 455–472.
[46] O. Tahri, P. R. Giordano, and Y. Mezouar, "Rotation free active vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep./Oct. 2015, pp. 3086–3091.

Eric Cristofalo (SM'15) received the B.S. degree in mechanical engineering from Drexel University, Philadelphia, PA, USA, in 2013, and the M.S. degree in mechanical engineering from Boston University, Boston, MA, USA, in 2016. He is currently pursuing the Ph.D. degree with the Multi-Robot Systems Lab, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, USA.

He is currently a National Defense Science and Engineering Graduate (NDSEG) Fellow with the Multi-Robot Systems Lab, Department of Aeronautics and Astronautics, Stanford University. His current research interests include vision-based control, active perception, and 3-D reconstruction with multirobot systems.

Eduardo Montijano (M'12) received the M.Sc. and Ph.D. degrees from the Universidad de Zaragoza, Zaragoza, Spain, in 2008 and 2012, respectively.

From 2012 to 2016, he was a Faculty Member with the Centro Universitario de la Defensa, Zaragoza. He is currently an Assistant Professor with the Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza. His current research interests include distributed algorithms, cooperative control, and computer vision.

Dr. Montijano received the Extraordinary Award from the Universidad de Zaragoza in 2012–2013 academic year for the Ph.D. degree.

Mac Schwager (M'04) received the B.S. degree from Stanford University, Stanford, CA, USA, in 2000, and the M.S. and Ph.D. degrees from MIT, Cambridge, MA, USA, in 2005 and 2009, respectively.

He was a Post-Doctoral Researcher with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA, and CSAIL, MIT, from 2010 to 2012. From 2012 to 2015, he was an Assistant Professor with Boston University, Boston, MA, USA. He is currently an Assistant Professor with the Aeronautics and Astronautics Department, Stanford University. His current research interests include distributed algorithms for control, perception, and learning in groups of robots and animals.

Dr. Schwager was a recipient of the NSF CAREER Award in 2014, the DARPA YFA in 2018, and a Google Faculty Research Award in 2018.