Optimal Visual Servoing for Differentially Flat Underactuated Systems

Matthew Sheckells, Gowtham Garimella, and Marin Kobilarov[1]

***Abstract*— This work introduces a hybrid visual servoing technique for differentially flat, underactuated systems that is well suited for aggressive dynamics. Standard Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS) approaches for underactuated systems, such as quadrotors, oftentimes do not explicitly ensure that the relevant image features stay in the camera's field of view, especially while the system is performing agile maneuvers. We present a control technique that is designed to mitigate this issue and that results in increased robustness. Given a goal image, we first solve a constrained Perspective-n-Point (PnP) problem to find an equilibrium pose which aligns the camera with the goal. We then formulate the task of navigating to the goal pose as an optimal control problem, where a cost over the resulting image feature tracks along the trajectory is minimized which implicitly keeps features in the field of view over the course of the trajectory. The optimization is performed over a polynomial parametrization of the flat outputs of the system to decrease the dimensionality of the optimization. Simulations and physical experiments are performed with a quadrotor system to benchmark the algorithm's performance against a typical PBVS approach.**

I. INTRODUCTION

Imagine a quadrotor equipped with a camera that is observing an object of interest, where it must use aggressive maneuvers to track the object. The quadrotor must accelerate quickly and move toward the target, potentially losing sight of the target as the quadrotor tilts. This type of motion would result in a system failure for a typical visual controller that does not take into account the complex interaction between vehicle dynamics and image features. In this paper, we consider a robust visual servo control technique that can perform agile maneuvers without losing track of important image features.

Visual servoing is concerned with using visual feedback in the control loop of a robot in order to increase the flexibility and accuracy of the robot's positioning system [1] and has been thoroughly studied over the past two decades [2], [3], [4]. Recent research has looked at applying visual servoing methods to quadrotor vehicles to perform complex tasks like autonomous landing [5], grasping and perching [6], and object tracking [7]. Visual servoing techniques are typically divided into two classes: Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS). PBVS estimates the goal position using visual information and uses standard state-space control methods to navigate to the goal. In contrast, IBVS formulates the control task directly in terms of image features. By steering the image features to a desired configuration, the Cartesian space motion-planning problem is implicitly solved [8]. It has been shown that both IBVS and PBVS strategies have their own weaknesses [9]. For instance, although the control design is straightforward in PBVS, the goal state solution is sensitive to camera calibration errors and to errors in the 3D environment model used to compute the goal position [1]. Notably, one of the drawbacks of PBVS is that it cannot guarantee that the target features stay in the field of view of the camera. Originally developed for kinematic systems, IBVS remains particularly challenging for underactuated dynamic systems since for such systems the relationship between control inputs and image features is often complex.

Fig. 1: Pictured is the modified Matrice quadrotor used for the experiments. The environment is the Johns Hopkins University campus.

Considerable research has been conducted in applying both PBVS approaches [10], [11] and IBVS techniques [12], [13], [14] to aerial vehicles. Since PBVS approaches only use visual information to compute a desired goal pose, the control strategy completely ignores feature locations when providing inputs to the system, so it is possible that the features could leave the camera's field of view. For quadrotors, certain ad-hoc approaches can be employed to mitigate the issue, like limiting the roll and pitch of the vehicle, but this severely limits the maneuverability of the aircraft. Oftentimes, IBVS approaches decouple the rotational kinematics of the vehicle from the image features by expressing the image feature error in a "rotation-compensated" camera frame whose roll and pitch are always zero [5], [7], [15], [6]. Although these works develop controllers that guarantee the image error in the rotation-compensated frame will converge to zero, it is still possible for the image features to completely leave the camera field of view if the system has significant rotation,

[1]Matthew Sheckells, Gowtham Garimella and Marin Kobilarov are with the Department of Computer Science and the Department of Mechanical Engineering, Johns Hopkins University, 3400 N Charles Str, Baltimore, MD 21218, USA msheckells|ggarime1|marin@jhu.edu

resulting in tracking failure. Bourquardez et. al. present several IBVS control techniques which decouple the image space from the task space by using spherical image moments as features [16]. Since the image error becomes a function of position only, large rotations could still occur, making the system vulnerable to failure as previously described. Ozawa et. al. present an IBVS method that does not neglect the rotational dynamics of the quadrotor [17]. It uses a virtual spring force to keep the quadrotor from rotating too much relative to a gravity-aligned frame. This technique was successful for the case considered, but in practice will limit the ability of the quadrotor to move aggressively.

Hybrid visual servoing techniques, such as *2-1/2-D visual servoing* [18], combine image features and 3D data to create a control law which is more stable and/or efficient than IBVS or PBVS alone. For instance, Hafez et. al. use a boosting technique to combine the outputs of an IBVS and PBVS controller to form a more robust hybrid controller [19], and Silveira et. al. use a Linear-Quadratic Regulator (LQR) controller to stabilize an airship, where the system model is augmented by the inclusion of image feature dynamics [20].

We propose a hybrid visual servoing technique for under-actuated systems based on an optimal control formulation of the objective behavior. The algorithm first solves a modified Perspective-n-Point (PnP) problem to find a relative goal state which minimizes the distance between point image features matched between the camera image and a goal set of features. With the final state fixed, a trajectory ending at the desired state is optimized to minimize a cost function which includes both control effort and image feature tracks. This special cost ensures that the image features remain in view of the camera so long as it is dynamically feasible for the system. The optimization is performed over a polynomial parametrization of the flat outputs of the system to increase efficiency. To solve the PnP problem, it is assumed that a depth map is available or can be estimated for the image stream, e.g. from a stereo camera, SLAM system, or Structure-from-Motion (SfM) algorithm.

Our approach differs from previous methods in several ways. First, unlike typical visual servoing techniques, the computed output is an entire dynamically feasible trajectory instead of an instantaneous control law. Second, the cost function used in the optimization-based formulation naturally ensures that enough features remain in the field of view even in the presence of aggressive dynamics. This makes the method suitable for tracking while performing highly agile maneuvers. The resulting computed trajectory can be tracked using a traditional non-linear controller. Experimental results show the feasibility of this approach in Section IV. Alternatively, the approach can be employed in a receding-horizon Model Predictive Control (MPC) framework where only the first few controls of the computed trajectory are executed and the entire trajectory is reoptimized at each time step [21]. Simulation experiments compare this method to a traditional PBVS approach in Section III, and validate its increased robustness.

The structure of the rest of the paper is as follows. The algorithm is first described in Section II. We then show the results of computer simulations of a quadrotor with a front-facing camera in Section III. Experimental results implemented on a real quadrotor are presented in Section IV, and a conclusion is given in Section V.

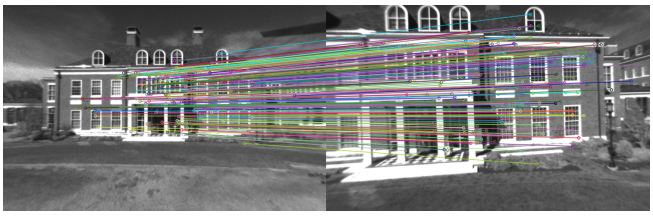


Fig. 2: Example feature matches between an initial image (left) and goal image (right).

II. Optimal Visual Servoing (OVS)

Consider a system with state $x(t)$ and control inputs $u(t)$ with dynamics given by $\dot{x} = f(x, u)$. The state space $X = SE(3) \times \mathbb{R}^6$ includes a Euclidean pose which for simplicity we assume coincides with the pose of a camera rigidly attached to the system. Given a goal image I_g and a camera image stream $I_c(t)$ at time t, feature sets are extracted from each image and denoted as $F_g = \{f_{g_1}, \ldots, f_{g_M}\}$ and $F_c(t) = \{f_{c_1}(t), \ldots, f_{c_M}(t)\}$, where f_{c_i} corresponds to f_{g_i}, respectively. The objective is to create a control law $u(F_g, F_c(t))$ that will minimize an error metric defined over the feature sets $e(F_g, F_c(t))$, thus aligning I_g and $I_c(t_f)$ at some final time t_f. Here, we assume that F_g and $F_c(t)$ are sets of point features in 2D image space. We take the error metric to be the Euclidean distance between features matched between F_g and $F_c(t)$.

The proposed algorithm for minimizing e consists of two steps. The first finds an equilibrium state of the system relative to its current pose that minimizes $e(F_g, F_c(t_f))$, denoted x_f. By equilibrium state, we mean x_f must be such that there exists a control u so that $f(x, u) = 0$. Thus, upon reaching x_f, the system should be able to maintain this state and keep the images aligned. The second step of the proposed method optimizes the trajectory from the current pose to the goal state x_f so that the image feature error along the trajectory is minimized, which implicitly forces the features to stay in the camera's field of view. We discuss the two steps in detail in the following sections.

A. Goal Pose Computation

In this step, we find an equilibrium state of the system x_f that will minimize the image error $e(F_g, F_c(t_f))$. First, SURF point descriptors [22] are extracted from I_g and $I_c(t)$ and matched. The matches are filtered using a ratio test as suggested by Lowe [23] to give the corresponding feature sets F_g and F_c (with perhaps false correspondences). An example set of feature matches is shown in Figure 2. Here, we assume that a depth map D can be obtained for $I_c(t)$ and that the intrinsic calibration parameters π (e.g. focal length and principal point) are known for the camera. D is used

to back-project the features in $I_c(t)$ to 3D world points to give a set of 2D-3D feature correspondences between the two images. We now denote the 3D features as $P_i \in \mathbb{R}^3$ and the corresponding 2D features as $p_i \in \mathbb{R}^2$. Finding the relative pose between two images with a set of 2D-3D feature correspondences, or the PnP problem, is well-studied [24]. Here, we use a Gauss-Newton optimization in a RANSAC framework [25] to minimize the reprojection error of the inlier features, subject to the constraint that the resulting pose must be an equilibrium point for the system. That is we solve

$$x_f = \arg\min_x \sum_{i \in \mathcal{I}} \|\mathcal{P}(x, P_i; \pi) - p_i\|^2$$
$$\text{s.t. } f(x, u) = 0 \text{ for some } u,$$

where $\mathcal{P} : X \times \mathbb{R}^3 \to \mathbb{R}^2$ is the standard pinhole camera projection function and $\mathcal{I}$ is the set of inlier features. To enforce the equilibrium constraint, we fix state variables to an equilibrium solution known *a priori* and only allow the solver to vary parts of the state for which the equilibrium is invariant. For the case of the quadrotor, we fix the roll, pitch, velocity, and angular velocity of the state to be zero during optimization and allow the solver to vary the position and yaw of the system to minimize the reprojection error.

This step alone is enough to perform typical PBVS, where the computed goal pose x_f can be used in a 3D control law. It does not, however, ensure that the image features stay in the field of view of the camera. Take, for example, a quadrotor with a fixed, downward-facing camera. If it is looking at a set of features on the ground plane and determines that it needs to move parallel to the plane to reduce the image error, then it must first tilt its frame (and, thus, the camera) in order to generate a force in that direction. For situations where a high degree of agility is required, the quadrotor may need to tilt significantly, causing the features to leave the camera's field of view. Without features in view, the controller would not be able to produce a new input to the system. In the next section, we describe the second step of the proposed algorithm which mitigates this issue.

B. Trajectory Optimization

The goal is to generate a discrete trajectory consisting of N segments $x_{0:N} \triangleq \{x_0, x_1, \ldots, x_N\}$ and a set of corresponding controls $u_{0:N-1} \triangleq \{u_0, u_1, \ldots, u_{N-1}\}$ for $u_i \in \mathbb{R}^m$ which minimize a cost $J(x_{0:N}, u_{0:N-1})$ subject to a chosen time-stepping rule of the dynamics expressed as $x_{k+1} = f_k(x_k, u_k)$. The time-step of the dynamics is given as t_f/N so that each x_i occurs at a time $t_i = it_f/N$, where t_f is a fixed final time chosen by the user. Although it is possible to optimize over t_f as part of the formulation, we manually choose it here for simplicity and so that the trajectory will have some nominal speed s_{nom} (i.e. we choose the final time as $t_f = \Delta p / s_{nom}$ where Δp is the distance from the start to the goal position). The standard approach is to express the cost as

$$J(x_{0:N}, u_{0:N-1}) = L_N(x_N) + \sum_{k=0}^{N-1} L_k(x_k, u_k),$$

where L_N is a terminal cost and L_k is a cost along the trajectory. We exclude L_N for the remainder of this work since we fix the final state in the trajectory. Typically, the control cost is defined as

$$L'_k(x_k, u_k) = \frac{1}{2}(x_k - x_f)^T Q_k (x_k - x_f) + \frac{1}{2} u_k^T R_k u_k,$$

where $Q_k \geq 0$ and $R_k > 0$ are matrices which define the control objective. However, we use an augmented cost L_k with terms over each image feature reprojection error

$$e(x_k)_i = \mathcal{P}(x_k, P_i; \pi) - p_i,$$

where $\mathcal{P} : X \times \mathbb{R}^3 \to \mathbb{R}^2$ is the camera projection function and π are the camera intrinsics. This gives

$$L_k(x_k, u_k) = L'_k(x_k, u_k) + \frac{1}{2} H_k \left[\frac{1}{M} \sum_{i=1}^{M} e_i(x_k)^\top e_i(x_k) \right]$$

as the cost along the trajectory, where $H_k > 0$ is a scalar weight. Since we include the feature errors at each step along the trajectory in J, an optimal solution which minimizes the cost will naturally reduce the reprojection error along the course of the trajectory, which will keep the desired set of features in view.

To optimize the trajectory, we use a well-established optimal control technique called direct shooting [26], which exploits the least-squares nature of J. In direct shooting, the discrete control trajectory is parametrized by a vector $\xi \in \mathbb{R}^{l \leq Nm}$, e.g. ξ could be control points of a polynomial spline. The parametrization ξ can be mapped back to the controls by using a function ϕ_k (which depends on the chosen parametrization) to give $u_k = \phi_k(\xi)$ for each $k = 0, \ldots, N-1$. Each state can be expressed as a function of ξ by using the dynamics, so we have $x_k = \eta_k(\xi)$. The cost is then expressed as $J(\xi) = \frac{1}{2} h(\xi)^T h(\xi)$, where $h : \mathbb{R}^l \to \mathbb{R}^{Nm + (N-1)(n+2M)}$ is given by

$$h(\xi) = \begin{bmatrix} \sqrt{R_0}\phi_0(\xi) \\ \sqrt{Q_1}(\eta_1(\xi) - x_f) \\ \sqrt{R_1}\phi_1(\xi) \\ \sqrt{H_1/M}e_1(\eta_1(\xi)) \\ \vdots \\ \sqrt{H_1/M}e_M(\eta_1(\xi)) \\ \vdots \\ \vdots \\ \sqrt{Q_{N-1}}(\eta_{N-1}(\xi) - x_f) \\ \sqrt{R_{N-1}}\phi_{N-1}(\xi) \\ \sqrt{H_{N-1}/M}e_1(\eta_{N-1}(\xi)) \\ \vdots \\ \sqrt{H_{N-1}/M}e_M(\eta_{N-1}(\xi)) \end{bmatrix}.$$

Since $R_i > 0$, the Jacobian $\partial h(\xi)$ is guaranteed to be full rank and one can apply a Gauss-Newton (GN) iterative method to optimize the trajectory. That is, we update $\xi \to \xi + \delta\xi$ where $\delta\xi = -(\partial h^T \partial h)^{-1} \partial h^T h$. A major advantage of the GN algorithm is its simplicity and robustness by using

standard regularization and line-search techniques [27]. Note that we overload $\pm$ to be a valid operator on $SE(3)$ as in [28], [29].

In this work, we actually parametrize the *flat outputs* of the trajectory instead of the controls. This allows the final state of the trajectory to be fixed during optimization, as explained in Section II-C.

C. Differential Flatness and Trajectory Parametrization

We next present an extension to the basic flatness approach which can handle arbitrary linear boundary constraints. A system with state $x \in \mathbb{R}^n$ and inputs $u \in \mathbb{R}^m$ is differentially flat if one can find outputs $y \in \mathbb{R}^m$ of the form $y = h(x, u, \dot{u}, \ldots, u^{(a)})$ and functions ψ and α such that $x = \psi(y, \dot{y}, \ldots, y^{(b)})$ and $u = \alpha(y, \dot{y}, \ldots, y^{(c)})$, where $y^{(a)}$ is taken to mean the a-th order derivative of y.

To reduce the number of optimization parameters, the trajectory is parametrized in time as a Bézier curve over the flat outputs $y(t)$ of the chosen system. This parametrization is infinitely differentiable which ensures continuity in the state trajectory $x(t)$. After optimizing $y(t)$, the states and controls necessary to produce the trajectory can be recovered using α and ψ.

The number of control points used to define the curve is n_k, the dimension of the flat output space is m, and the control points are denoted by $k_i \in \mathbb{R}^m$ for $i = 1, \ldots, n_k$. The Bézier basis for the i-th control point is given by

$$B_i = \binom{n_k}{i}(1-t)^{n_k - i}t^i.$$

The flat outputs are then given by

$$y(t) = \sum_{i=0}^{n_k} B_i k_i.$$

The derivatives of the Bézier basis can be defined recursively in terms of the basis itself.

To fix the initial and final states of the trajectory, the first b derivatives of $y(t)$ must be fixed at times $t = 0$ and $t = t_f$. The fixed conditions of the trajectory can be described by a linear system of equations $\boldsymbol{BK} = \boldsymbol{Y}$ defined by

$$\underbrace{\begin{bmatrix} B_1(0) & \cdots & B_{n_k}(0) \\ B_1'(0) & \cdots & B_{n_k}'(0) \\ \vdots & & \vdots \\ B_1^{(b)}(0) & \cdots & B_{n_k}^{(b)}(0) \\ B_1(t_f) & \cdots & B_{n_k}(t_f) \\ B_1'(t_f) & \cdots & B_{n_k}'(t_f) \\ \vdots & & \vdots \\ B_1^{(b)}(t_f) & \cdots & B_{n_k}^{(b)}(t_f) \end{bmatrix}}_{\boldsymbol{B}} \underbrace{\begin{bmatrix} k_1^T \\ k_2^T \\ \vdots \\ k_{n_k}^T \end{bmatrix}}_{\boldsymbol{K}} = \underbrace{\begin{bmatrix} y^T(0) \\ y'^T(0) \\ \vdots \\ y^{(b)^T}(0) \\ y^T(t_f) \\ y'^T(t_f) \\ \vdots \\ y^{(b)^T}(t_f) \end{bmatrix}}_{\boldsymbol{Y}}. \tag{1}$$

We require $n_k \geq 2(b+1)$ so that at least one solution exists for $\boldsymbol{K}$. In this approach, we choose $n_k > 2(b+1)$ so that the system is underdetermined and yields infinitely many solutions. We find a particular solution $\boldsymbol{K}_R$ with $\boldsymbol{K}_R = \boldsymbol{B}^\dagger \boldsymbol{Y}$, where $\boldsymbol{B}^\dagger$ is the Moore-Penrose pseudoinverse of $\boldsymbol{B}$ and the subscript R denotes that fact that the solution is in the range of $\boldsymbol{B}$. Note that if $\boldsymbol{K}_N$ is in the nullspace of $\boldsymbol{B}$, then $\boldsymbol{K}_N + \boldsymbol{K}_R$ also satisfies Equation 1. So, we can optimize over $\boldsymbol{K}_N$ while guaranteeing that the boundary conditions of the resulting trajectory will always be satisfied. One can see that for $n_k > 2(b+1)$, the rank of $\boldsymbol{B}$ is at most $2(b+1)$. Thus, the dimension of its nullspace is at least $n_k - 2(b+1)$ by the rank-nullity theorem. A basis $\boldsymbol{B_N}$ for the nullspace of $\boldsymbol{B}$ can be found by numerically computing its singular value decomposition (where $\boldsymbol{B} = \boldsymbol{U\Sigma V}^T$) and then taking $\boldsymbol{B_N} = [V_{2b+2} \cdots V_{n_k}]$, where V_i is the ith column of $\boldsymbol{V}$.

For a matrix of weights $\boldsymbol{S} \in \mathbb{R}^{n_k - 2(b+1) \times m}$, a trajectory parametrization which satisfies the boundary conditions can be given as $\boldsymbol{K} = \boldsymbol{K}_N + \boldsymbol{K}_R = \boldsymbol{B}_N \boldsymbol{S} + \boldsymbol{K}_R$. The objective then is to find $\boldsymbol{S}$ which minimizes J, so for our GN formulation we have $\xi = vec(\boldsymbol{S})$, where $vec(\cdot)$ denotes the vectorization of the input matrix by stacking its columns into a single column. Once the optimal null space weights $\boldsymbol{S}^*$ are obtained, the optimal Bézier knots are given as $\boldsymbol{K}^* = \boldsymbol{B}_N \boldsymbol{S}^* + \boldsymbol{K}_R$. The i-th trajectory segment can then be recovered as

$$x_i = \psi\left(\sum_{j=1}^{n_k} B_j(t_i)k_j^*, \sum_{j=1}^{n_k} B_j'(t_i)k_j^*, \ldots, \sum_{j=1}^{n_k} B_j^{(b)}(t_i)k_j^*\right)$$

$$u_i = \alpha\left(\sum_{j=1}^{n_k} B_j(t_i)k_j^*, \sum_{j=1}^{n_k} B_j'(t_i)k_j^*, \ldots, \sum_{j=1}^{n_k} B_j^{(c)}(t_i)k_j^*\right),$$

where k_j^* is the j-th row of $\boldsymbol{K}^*$.

In summary, we have proposed a parametrization which automatically produces dynamically feasible boundary conditions matching the initial and desired states exactly and that can be further optimized in view of the cost function.

D. Trajectory Initialization

In practice, we initialize $\boldsymbol{S}$ from an existing trajectory $x_{init}(t), u_{init}(t)$, e.g. a straight line to the goal or a previously computed optimal trajectory. We do this by solving a weighted least-squares problem. First, we convert $x_{init}(t), u_{init}(t)$ into flat output space $y_{init}(t)$. Then for time samples $t_0, \ldots, t_f$ we setup a linear system of equations $\boldsymbol{AK} = \boldsymbol{Y}_{init}$ defined by

$$\underbrace{\begin{bmatrix} B_1(t_0) & \cdots & B_{n_k}(t_0) \\ B_1(t_1) & \cdots & B_{n_k}(t_1) \\ \vdots & & \vdots \\ B_1(t_f) & \cdots & B_{n_k}(t_f) \\ B_1^{(d)}(t_0) & \cdots & B_{n_k}^{(d)}(t_0) \\ B_1^{(d)}(t_1) & \cdots & B_{n_k}^{(d)}(t_1) \\ \vdots & & \vdots \\ B_1^{(d)}(t_f) & \cdots & B_{n_k}^{(d)}(t_f) \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} k_1^T \\ k_2^T \\ \vdots \\ k_{n_k}^T \end{bmatrix}}_{\boldsymbol{K}} = \underbrace{\begin{bmatrix} y^T(t_0) \\ y^T(t_1) \\ \vdots \\ y^T(t_f) \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{\boldsymbol{Y}_{init}}, \tag{2}$$

where d is the d-th derivative of the flat outputs that we would like to minimize in addition to fitting the given trajectory. For the case of the quadrotor, we have $d = 4$ to

minimize the snap of the resulting trajectory, which produces a smooth control response. We can write Equation 2 as

$$\boldsymbol{AK} = \boldsymbol{A}(\boldsymbol{K}_R + \boldsymbol{K}_N) = \boldsymbol{A}(\boldsymbol{B}_N\boldsymbol{S} + \boldsymbol{K}_R) = \boldsymbol{Y}_{init}.$$

We then solve for $\boldsymbol{S}$ using weighted least-squares as $\boldsymbol{S} = [(\boldsymbol{AB}_N)^T\boldsymbol{W}(\boldsymbol{AB}_N)]^{-1}(\boldsymbol{AB}_N)^T\boldsymbol{W}(\boldsymbol{Y}_{init} - \boldsymbol{AK}_R)$, where $\boldsymbol{W}$ is a matrix of weights that determines the tradeoff between smoothness and fitting the given trajectory.

III. Model Predictive Control Simulation

We next analyze the performance of the algorithm described in Section II to control a quadrotor in a MPC framework. That is, we fly the quadrotor with the controls computed by the algorithm for a small duration of time and then regenerate the optimal trajectory from the resulting state of the quadrotor. This is repeated until convergence. We compare our method to a PBVS approach which finds the position of the goal image relative to the current position and then computes an optimal trajectory to the goal state which minimizes control effort, i.e. we find controls which minimize J with $H_k = 0$. This is also done in a MPC framework. Thus, the methods compared are identical except for the introduction of feature error terms in J for the proposed method, which allows for a fair evaluation of the algorithm.

A. System

The system simulated is a simplified quadrotor model with a front-facing camera. It takes four controls as input: three torques that correspond to the body x, y, and z axes $\tau = (\tau_1, \tau_2, \tau_3)$ and a thrust force T along the z-axis of the quadrotor. It is also subject to gravity. This system has a state $x = (p, R, v, \omega) \in \mathrm{SE}(3) \times \mathbb{R}^6$ and is known to be differentially flat [30]. The dynamics are given as

$$\begin{aligned}
\dot{p} &= v, \\
\dot{v} &= \frac{1}{m}Re_3T + g + \frac{w}{m}, \\
\dot{R} &= R\hat{w}, \\
\dot{w} &= \mathbb{J}^{-1}[\mathbb{J}\omega \times \omega + \tau],
\end{aligned}$$

where g is the gravity vector, $\mathbb{J}$ is the inertia tensor, m is the mass of the quadrotor, and $w \sim N(0, I\sigma^2)$ is additive Gaussian noise corresponding to external disturbance forces.

The flat outputs are given as $y = (p_1, p_2, p_3, \gamma) \in \mathbb{R}^4$, where γ is the yaw of a roll-pitch-yaw parametrization of the rotation matrix R. Denote $y_t = (y_1, y_2, y_3)$. We recover the full state and controls from the flat outputs using the following relations: $p = y_t$, $\dot{p} = \dot{y}_t$, $T = \|m(\ddot{y}_t - g)\|$ and the three columns of the rotation matrix R_x, R_y, R_z are reconstructed as

$$\begin{aligned}
R_z &= m(\ddot{y}_t - g)/T, \\
R_y &= R_z \times \begin{pmatrix} \cos y_4 \\ \sin y_4 \\ 0 \end{pmatrix} \Big/ \left\| R_z \times \begin{pmatrix} \cos y_4 \\ \sin y_4 \\ 0 \end{pmatrix} \right\|, \\
R_x &= R_y \times R_z.
\end{aligned}$$

Fig. 3: The simulated quadrotor flies in the urban environment above. The blue line illustrates an example optimal trajectory. The image on the lower right corner shows the goal image for the simulation experiments.

The angular velocity is recovered as

$$\begin{aligned}
\omega_x &= -R_y \cdot \dddot{y}_t/T, \\
\omega_y &= R_x \cdot \dddot{y}_t/T, \\
\omega_z &= \dot{y}_4(e_3 \cdot R_z).
\end{aligned}$$

as derived in [31], where e_3 is the standard unit vector along the z-axis. To recover τ, we first recover $\dot{\omega}$. Note that from the dynamics

$$my_t^{(4)} = (R\hat{\omega}^2 + R\hat{\dot{\omega}})Te_3 + 2R\hat{\omega}\dot{T}e_3 + R\ddot{T}e_3.$$

Solving this for $\dot{\omega}$ gives

$$\begin{aligned}
\dot{\omega}_x &= (-mR_y \cdot y_t^{(4)} - w_y w_z T + 2w_x\dot{T})/T \\
\dot{\omega}_y &= (mR_y \cdot y_t^{(4)} - w_x w_z T - 2w_y\dot{T})/T \\
\dot{\omega}_z &= \ddot{y}_4 e_3 \cdot R_z + \dot{y}_4 e_3^T R\hat{\omega}e_3.
\end{aligned}$$

Then, we use the dynamics $\tau = \mathbb{J}\dot{\omega} - \mathbb{J}\omega \times \omega$ and τ is recovered.

B. Virtual Reality Setup using Real Outdoor 3D Data

We execute the proposed method and the PBVS method given the same goal image and starting from the same initial state for multiple trials. We vary the nominal speed by adjusting t_f across trials to evaluate how well the methods perform during more aggressive dynamics.

The workspace environment considered is a small section of the Johns Hopkins University campus. A to-scale 3D model of the area was created using the Agisoft Photoscan SfM software [32].

The initial attitude of the system was always specified to have zero roll and pitch. When performing the first stage of the algorithm, we constrained the optimization to only consider final states with zero roll and pitch with $v = 0$ and $\omega = 0$. The quadrotor starts at a static position about 8m away from the goal pose, which is not known beforehand. The goal image and a picture illustrating the setup are shown in Figure 3.

Fig. 4: Sequence of onboard camera images from the MPC simulations for OVS (top) and PBVS (bottom) over the first 0.4s of the trajectory. The green circle marks a prominent image feature in the field of view. Note that, in contrast to the OVS case, the image feature is close to exiting the camera's field of view during execution of the PBVS controller.

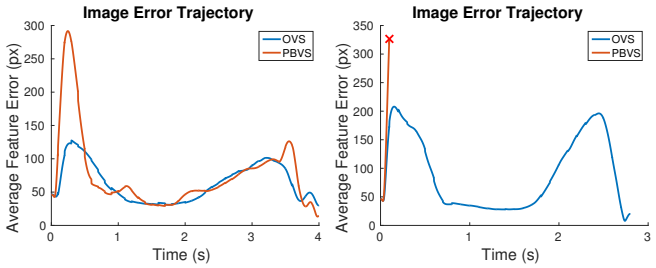


Fig. 5: Comparison of simulated MPC PBVS and OVS trajectories for $s_{nom} = 2.07$m/s (left) and $s_{nom} = 2.95$m/s (right). A red 'x' marks a system failure.

		Avg. Pixel Error		Max Pixel Error	
t_f (s)	s_{nom} (m/s)	OVS	PBVS	OVS	PBVS
4.0	2.07	63.8	75.8	127.6	291.6
3.8	2.17	68.0	90.9	130.7	192.8
3.6	2.30	71.6	104.4	155.4	202.9
3.4	2.43	74.1	105.3	157.4	202.9
3.2	2.58	77.1	128.1	159.8	265.2
3.0	2.75	85.3	Failure	195.1	Failure
2.8	2.95	90.7	Failure	207.9	Failure

TABLE I: Comparison of simulated image error trajectory statistics for each method across different nominal speeds.

The Bézier curve parametrization in our approach used $n_k = 20$ control points to provide enough degrees of freedom for shaping the trajectory. We used $N = 64$ trajectory segments in the optimization. The GN trajectory optimization used numerically calculated gradients. We simulated MPC at a rate of 10Hz with the underlying controls being executed at 500Hz. Gaussian noise was added to the dynamics to simulate unknown external forces with $\sigma = 1$N. Since ground truth feature matches are not known, the system must deal with possibly false matches that are not filtered out by the steps described in Section II-A. Measurements of rotation are assumed to be perfect since an IMU can provide an accurate estimate on a real system. The depth map of the features is also assumed to be perfect in this simulation, though in practice this is not always the case.

C. Results

Image error trajectories for the OVS and PBVS methods for various values of t_f are shown in Figure 6b. Quantitative measures of the image error trajectories are given in Table I. Notably, for the proposed method, the maximum feature error never exceeds 210 pixels and the average feature errors and max pixel errors are well below those of the PBVS approach in all cases. Furthermore, the PBVS system totally fails due to lack of features for maneuvers with nominal speeds greater than 2.75m/s, whereas the proposed method succeeds. However, the reduced image error of OVS comes with a performance trade-off. The RMS thrust of each OVS trajectory is noticeably larger than that of the corresponding PBVS trajectories as shown in Table II. The OVS RMS torque is actually reduced compared to the PBVS values in this situation. Putting a cost on image feature error reduces the overall rotation of the system, which results in smaller angular accelerations. The tuning parameters H_k, R_k can be adjusted by the user to obtain the desired trade-off between control effort and image error.

Example position space trajectories for the two methods are shown in Figure 6a. Qualitatively, the OVS trajectory begins by ascending so that when it pitches forward the goal features will remain in view. The trajectory also dips slightly at the end so that the features remain in view when it pitches backward to slow down. The PBVS trajectory has to pitch forward significantly in the beginning in order to move as fast as the OVS trajectory. This causes many features to move out of or near the edge of the camera's field of view as depicted in Figure 4, increasing the feature error relative to OVS as shown in an overlaid comparison of image error trajectories given in Figure 5. The OVS trajectory also has a significantly larger curvature compared to that of PBVS. In the current formulation this could make it more difficult to navigate in a cluttered environment among obstacles. However, state constraints can be imposed on the optimization which guarantee an obstacle-free trajectory or additional cost terms can be added to penalize intersection with obstacles. This is left for future work.

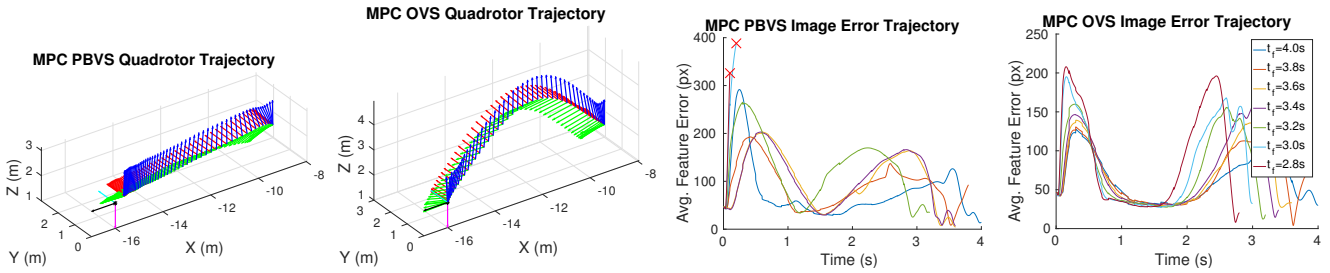
IV. Trajectory Tracking Experiment

A. Experimental Setup

Our prototype platform is a DJI Matrice quadrotor with the DJI Guidance sensor suite. We use an onboard Intel NUC computer for processing. A picture of the platform is shown in Figure 1. Deph estimation of the environment is performed

		RMS Torque (Nm)		RMS Thrust (N)	
t_f (s)	s_{nom} (m/s)	OVS	PBVS	OVS	PBVS
4.0	2.07	0.0087	0.0213	5.27	5.06
3.8	2.17	0.0105	0.0183	5.36	5.13
3.6	2.30	0.0189	0.0119	5.47	5.16
3.4	2.43	0.0128	0.0209	5.60	5.18
3.2	2.58	0.0276	0.0300	5.84	5.33
3.0	2.75	0.0323	Failure	5.9215	Failure
2.8	2.95	0.0322	Failure	6.1866	Failure

TABLE II: Comparison of control effort statistics from simulated trajectories for each method across different nominal speeds.



(a) (b)

Fig. 6: The resulting simulated 3D trajectories from PBVS and OVS with $s_{nom} = 2.07$m/s are shown in (a). Shown in (b) are resulting image error trajectories for PBVS and OVS for several different final times. A red 'x' marks a system failure.

Fig. 7: Top shows the quadrotor following an OVS trajectory using a velocity controller. Bottom shows the quadrotor following a PBVS trajectory using the same method.

using a stereo camera pair. The depth map is passed to our optimal visual servoing algorithm described in Section II.

As direct torque and thrust controls are not exposed to the user, we control the quadrotor using velocity commands. Instead of using MPC, the optimal trajectory is computed all at once before motion begins and is then tracked using a velocity controller with position feedback, where the local position of the quadrotor is estimated using onboard fusion of GPS, accelerometer, and velocity measurements (from Guidance).

A series of four trajectories is executed for each control strategy with varying nominal speeds. The quadrotor starts from the same initial state each time and is given the same goal image each time. The distance from start to goal is about 11m, and the environment is an outdoor university campus.

Note that the goal position is not known beforehand and is computed from the goal image using image features as decribed in Section II-A. The trajectory optimization can be run at a rate of 10Hz on an i7 processor, so it is possible in practice to perform OVS in a MPC framework; however, we leave that demonstration for future work.

B. Results

Qualitative trajectories are given in Figure 7 and an example initial image and goal image are shown in Figure 2. Notice that the shape of the trajectories matches those

		Avg. Pixel Error	
t_f (s)	s_{nom} (m/s)	OVS	PBVS
6.0	1.80	71.2	100.2
5.0	2.20	97.1	122.8
4.0	2.75	128.6	143.1
3.5	3.10	184.9	200.8

TABLE III: Comparison of average image error for each method across different nominal speeds.

given in simulation. Table III gives the average image error over each trajectory for each method. Note that the average image error for OVS is consistently lower than that of the PBVS method, demonstrating the robustness of the proposed algorithm when implemented on a real system. Both OVS and PBVS use a noisy depth map to help compute the goal pose, leading to a noisy goal computation which results in a non-zero final image error in both methods even though the quadrotor ends up close to the goal state. In practice, the quadrotor is not able to perfectly follow the computed optimal trajectory, and the feedback controls for tracking the trajectory are not computed to be optimal with respect to the given cost (as they would be when using MPC). This accounts for the increase in image error for OVS relative to PBVS in certain parts of the image error trajectory as shown in Figure 8.

V. Conclusion

This work considers a hybrid visual servoing technique that is applicable to any differentially flat system and that is well suited for aggressive dynamics. The method formulates visual servoing in an optimal control context and uses a direct shooting method to minimize an image-based cost function. The trajectories are parametrized in flat output space using Bézier curves to reduce the dimensionality of the optimization problem. It has been shown experimentally that this method shows increased performance when compared to a traditional PBVS approach when applied to both simulated and real quadrotor systems. It is capable of keeping the target features in the view of the camera, even while flying highly dynamic trajectories. The is expected to be especially advantagous for tracking moving objects, which is left for future work. Additional future research will consider performance in the presence of image noise and will take obstacles into account during trajectory optimization.

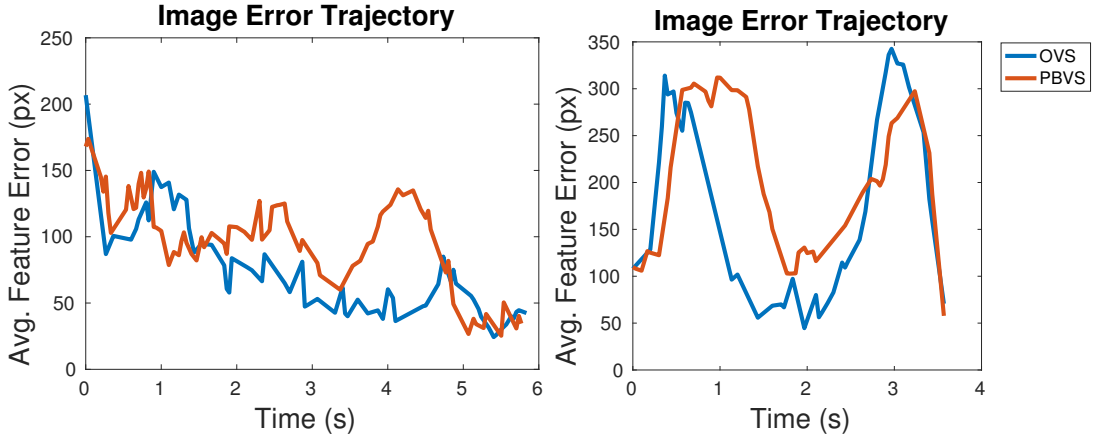


Fig. 8: A comparison of the resulting image error trajectories for the OVS and PBVS approach for a $s_{nom} = 1.8$m/s trajectory (left) and $s_{nom} = 3.1$m/s trajectory (right).

REFERENCES

[1] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[2] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, 1992.

[3] R. Pissard-Gibollet and P. Rives, "Applying visual servoing techniques to control a mobile hand-eye system," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1. IEEE, 1995, pp. 166–171.

[4] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE Journal of Robotics and Automation*, vol. 3, no. 5, pp. 404–417, 1987.

[5] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a vtol uav on a moving platform using image-based visual servoing," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 971–976.

[6] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward image based visual servoing for aerial grasping and perching," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2113–2118.

[7] S. Azrad, F. Kendoul, and K. Nonami, "Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm," *Journal of System Design and Dynamics*, vol. 4, no. 2, pp. 255–268, 2010.

[8] C. Samson, B. Espiau, and M. L. Borgne, *Robot Control: The Task Function Approach*. Oxford University Press, 1991.

[9] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The confluence of vision and control*. Springer, 1998, pp. 66–78.

[10] E. Altuğ, J. P. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback," in *IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 2002, pp. 72–77.

[11] L. Mejıas, S. Saripalli, P. Campoy, and G. S. Sukhatme, "Visual servoing for tracking features in urban areas using an autonomous helicopter," *Journal of Field Robotics*, vol. 23, pp. 195–199, 2006.

[12] T. Hamel and R. Mahony, "Visual servoing of an under-actuated dynamic rigid-body system: an image-based approach," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 187–198, 2002.

[13] N. Guenard, T. Hamel, and R. Mahony, "A practical visual servo control for an unmanned aerial vehicle," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 331–340, 2008.

[14] R. Mahony and T. Hamel, "Image-based visual servo control of aerial robotic systems using linear image features," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 227–239, 2005.

[15] H. Jabbari, G. Oriolo, and H. Bolandi, "Dynamic ibvs control of an underactuated uav," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2012, pp. 1158–1163.

[16] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck, "Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle," *IEEE Trans. on Robotics*, vol. 25, no. 3, pp. 743–749, 2009.

[17] R. Ozawa and F. Chaumette, "Dynamic visual servoing with image moments for a quadrotor using a virtual spring approach," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 5670–5676.

[18] E. Malis, F. Chaumette, and S. Boudet, "2 1/2d visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, 1999.

[19] A. A. Hafez, E. Cervera, and C. Jawahar, "Hybrid visual servoing by boosting ibvs and pbvs," in *International Conference on Information and Communication Technologies: From Theory to Applications*. IEEE, 2008, pp. 1–6.

[20] G. F. Silveira, J. R. H. Carvalho, P. Rives, J. R. Azinheira, S. S. Bueno, and M. K. Madrid, "Optimal visual servoed guidance of outdoor autonomous robotic airships," in *Proceedings of the American Control Conference*, vol. 1. IEEE, 2002, pp. 779–784.

[21] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[22] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[24] T. Petersen, "A comparison of 2d-3d pose estimation methods," *Master's thesis, Aalborg University-Institute for Media Technology Computer Vision and Graphics, Lautrupvang*, vol. 15, p. 2750, 2008.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[26] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[27] D. P. Bertsekas, "Nonlinear programming," 1999.

[28] M. Kobilarov, "Discrete optimal control on lie groups and applications to robotic vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5523–5529.

[29] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4692–4697.

[30] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 2520–2525.

[31] M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Inversion based direct position control and trajectory following for micro aerial vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 2933–2939.

[32] "Agisoft photoscan," http://www.agisoft.com.