Image-Based Trajectory Tracking Through Unknown Environments Without Absolute Positioning

Shiyu Feng, *Graduate Student Member, IEEE*, Zixuan Wu, Yipu Zhao, *Member, IEEE*, and Patricio A. Vela, *Member, IEEE*

***Abstract*—This article describes a stereo image-based visual servoing (VS) system for trajectory tracking by a nonholonomic robot without externally derived pose information nor a known visual map of the environment. It is called *trajectory servoing (TS)*. The critical component is a feature-based, indirect simultaneous localization and mapping (SLAM) method to provide a pool of available features with estimated depth, so that they may be propagated forward in time to generate image feature trajectories for VS. Short and long distance experiments show the benefits of TS for navigating unknown areas without absolute positioning. Empirically, TS has better trajectory tracking performance than pose-based feedback when both rely on the same underlying SLAM system.**

***Index Terms*—Computer vision, robot control, robotics.**

I. Introduction

NAVIGATION systems with real-time needs often employ hierarchical schemes that decompose navigation across multiple spatial and temporal scales. Doing so, improves real-time responsiveness to novel information gained from sensors, while being guided by the more slowly evolving global path. At the lowest level of the hierarchy lies trajectory tracking to realize the planned paths or synthesized trajectories. In the absence of an absolute reference (such as GPS) and of an accurate map of the environment, only on-board mechanisms can support trajectory-tracking. These include odometry through *proprioceptive* sensors (wheel encoders, IMUs, etc.) or visual sensors. Pose estimation from proprioceptive sensors is not observable, thus visual sensors provide the best mechanism to anchor the robot's pose estimate to external, static reference locations. Indeed visual odometry (VO) or visual SLAM (V-SLAM) solutions are essential in these circumstances. However, VO/V-SLAM estimation error and pose drift degrade trajectory tracking performance. Is it possible to do better?

The hypothesis explored in this article is that *performing trajectory tracking in the image domain reduces the trajectory tracking error of systems reliant on VO or V-SLAM pose estimates for feedback*. Trajectory tracking feedback shifts from pose space to perception space. Perception space approaches have several favorable properties when used for navigation [1], [2]. Shifting the representation from being world-centric to being viewer-centric reduces computational demands and improves run-time properties. For trajectory tracking without reliable absolute pose information, simplifying the feedback pathway by skipping processes that degrade performance of—or are not essential to—the local tracking task may have positive benefits. Using image measurements to control motion relative to visual landmarks is known as *visual servoing (VS)*. Thus, the objective is to explore the use of image-based VS for long distance trajectory tracking with a stereo camera as the primary sensor and without absolute positioning. The technique, which we call *trajectory servoing (TS)*, will be shown to improve trajectory tracking over systems reliant on V-SLAM for pose-based feedback. In this article, a trajectory is a time parametrized curve in Cartesian space.

A. Related Work

1) VS: VS has a rich history and a diverse set of strategies for stabilizing a camera to a target pose described visually. VS algorithms fit into one of two categories: image-based VS (IBVS) and position-based VS (PBVS) [3], [4]. IBVS implementations include both feature stabilization and feature trajectory tracking [4], [5]. IBVS emphasizes point-to-point reconfiguration given a terminal image state [6]. It requires artificial markers and co-visibility of image measurements during motion [7], which may not be satisfied for long-distance displacements. Furthermore, there is no guarantee on the path taken since the feature space trajectory has a nonlinear relationship with the Cartesian space trajectory. Identifying a feature path to track

Manuscript received 20 January 2022; revised 26 March 2022; accepted 2 May 2022. Date of publication 3 June 2022; date of current version 16 August 2022. Recommended by Technical Editor H. Liu and Senior Editor X. Chen. This work was supported in part by NSF Award under Grant 1849333. *(Shiyu Feng and Zixuan Wu equally contribution to this work.) (Corresponding author: Shiyu Feng.)*

Shiyu Feng is with the School of Mechanical Engineering, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30308 USA (e-mail: shiyufeng@gatech.edu).

Zixuan Wu and Patricio A. Vela are with the School of Electrical and Computer Engineering, Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30308 USA (e-mail: zwu380@gatech.edu; pvela@gatech.edu).

Yipu Zhao is with the Meta Reality Lab, Redmond, WA 98052 USA (e-mail: yipu.zhao@gatech.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TMECH.2022.3175819.

Digital Object Identifier 10.1109/TMECH.2022.3175819

based on a target Cartesian space trajectory requires mapping the target positions into the image frame over time to generate the feature trajectory [5].

The target application is trajectory tracking for a mobile robot with IBVS. Both holonomic [8] and nonholonomic [9]–[14] mobile robots have been studied as candidates for VS. Most approaches do not use the full IBVS equations involving the image Jacobian. The centroid of the features [9], [13], the most frequent horizontal displacement of the matched feature pairs [10] or other qualitative cost functions [8] are used to generate [9] or correct [10] the feedforward angular velocity of mobile robot. These simplifications work well in circumstances tolerant to high tracking inaccuracy (e.g., an outdoor, open field navigation). In this article, we use more precise velocity relations between the robot and feature motion to generate a feedback control signal for exact tracking similar to [11]. That work studied the path following problem with a visible path marker line, which does not hold here.

2) Visual Teach and Repeat: Evidence that visual features can support trajectory tracking or consistent navigation through space lies in the *Visual Teach and Repeat* (VTR) navigation problem in robotics [9], [10]. Given data or recordings of prior paths through an environment, robots can reliably retrace past trajectories. The teaching phase of VTR creates a visual map that contains features associated with robot poses obtained from visual odometry [9], [15]–[18]. Extensions include real-time construction of the VTR data structure during the teaching process, and the maintenance and updating of the VTR data during repeat runs [15], [16]. Feature descriptor improvements make the feature matching more robust to the environment changes [18], [19]. Visual data in the form of feature points can have task relevant and irrelevant features, which provide VTR algorithms an opportunity to select a subset that best contributes to the localization or path following task [15], [17]. It is difficult to construct or update a visual map in real-time while in motion due to the separation of the teach and repeat phases. In addition, VTR focuses more on local map consistency and does not work toward global pose estimation [17] compared with SLAM since the navigation problems it solves are usually defined in the local frame.

Another type of VTR uses the optical flow [10], [20] or feature sequences [21]–[23] along the trajectory, which is then encoded into a VTR data structure and control algorithm in the teaching phase. Although similar to VS, the system is largely overdetermined. It can tolerate feature tracking failure, compared with traditional visual servo system, but may lead to discontinuities [24]. Though this method handles long trajectories, and may be supplemented from new teach recordings, it can only track paths through visited space.

3) Navigation Using Visual SLAM: Visual simultaneous localization and mapping (V-SLAM) systems estimate the robot's trajectory and world structure as the robot moves through space [25]. SLAM derived pose estimation naturally supports navigation or path following [26], [27], and PBVS [28], [29]. Most of these methods still need to initialize and maintain topological or metric visual maps for path planning [26], [27] or for localizing the robot for PBVS [28], [29]. These works do not involve Cartesian trajectories [26], [28], [29] nor provide solutions to occlusion and tracking loss problems [26]–[29]. The shortcomings imply the inability to track long Cartesian trajectories through unknown environments.

Pose estimation accuracy of SLAM is a major area of study [30]–[34]. However, most studies only test under open-loop conditions [30]–[32], [34], i.e., they analyze the pose estimation difference with the ground truth trajectory and do not consider the error induced when the estimated pose informs feedback control. More recently, closed-loop evaluation of V-SLAM algorithms as part of trajectory tracking feedback control or navigation were tested for individual SLAM systems [35]–[37] or across different systems [38]. The studies exposed the influence of V-SLAM estimation drift and latency on pose-based feedback control performance. This article builds upon an existing closed-loop benchmarking framework [38] and shows improved tracking performance by TS.

B. Contribution

Both IBVS and VTR depend on reliably tracked, known features within the field-of-view, which can only be achieved by artificial, task-oriented scenarios [5], [11] or a prebuilt trajectory map [9], [15]–[18]. This limitation does not permit travel through unknown environments to an unseen terminal state, and motivates the use of feature-based V-SLAM systems with online map saving [26], [27]. Compared with PBVS based on SLAM [28], [29], TS uses IBVS to less frequently query the pose from a stereo V-SLAM system [30], thereby attenuating the impact of estimation drift and error on trajectory tracking performance.

TS bypasses the explicit use of VO/V-SLAM pose, whose estimates are vulnerable to image-driven uncertainty, which manifests as pose error or drift [3], [4], by relying on the V-SLAM feature maintenance components that provide accurate and robust feature tracking and mapping. We present evidence for the assertion that the coupling between V-SLAM and IBVS combines their advantages to provide more effective feedback signals relative to V-SLAM pose-based control. Simulation and real experiment performance benchmarking show that TS improves trajectory tracking performance over pose-based feedback using SLAM estimates, thereby mitigating the effect of pose estimation error or drift, even though the same visual information is used for closing the loop. The beneficial coupling is the motivation behind the TS system design. It is a promising approach to trajectory tracking through unknown environments in the absence of absolute positioning signals.

C. Image-Based VS Rate Equations

The core algorithm builds on IBVS [39]. This section covers IBVS with an emphasis on how it relates the velocity of image features to the robot velocities via the image Jacobian [3], [4]. These equations will inform the trajectory tracking problem under nonholonomic robot motion. We use the more modern notation from geometric mechanics [40] since it provides equations that better connect to contemporary geometric control and to SLAM formulations for moving rigid bodies.

1) Nonholonomic Robot and Camera Kinematic Models: Let the motion model of the robot be a kinematic Hilare robot model, where the pose state $g_{\mathcal{R}}^{\mathcal{W}} \in SE(2)$ evolves under the control $\boldsymbol{u} = [\nu, \omega]^T$ as

$$\dot{g}_{\mathcal{R}}^{\mathcal{W}} = g_{\mathcal{R}}^{\mathcal{W}} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu \\ \omega \end{bmatrix} = g_{\mathcal{R}}^{\mathcal{W}} \cdot \xi_{\boldsymbol{u}} \tag{1}$$

for ν a forward linear velocity and ω an angular velocity, and $\xi_{\boldsymbol{u}} \in \mathfrak{se}(2)$. The state is the robot frame $\mathcal{R}$ relative to the world frame $\mathcal{W}$. The camera frame $\mathcal{C}$ is presumed to be described as $h_{\mathcal{C}}^{\mathcal{R}}$ relative to the robot frame. Consequently, camera kinematics relative to the world frame are

$$\dot{h}_{\mathcal{C}}^{\mathcal{W}} = g_{\mathcal{R}}^{\mathcal{W}} \cdot h_{\mathcal{C}}^{\mathcal{R}} \cdot \mathrm{Ad}_{h_{\mathcal{C}}^{\mathcal{R}}}^{-1} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu \\ \omega \end{bmatrix} = g_{\mathcal{R}}^{\mathcal{W}} \cdot h_{\mathcal{C}}^{\mathcal{R}} \cdot \zeta_{\boldsymbol{u}} \tag{2}$$

with $\zeta_{\boldsymbol{u}} \in \mathfrak{se}(2)$. Now, let the camera projection equations be given by the function $\boldsymbol{H} : \mathbb{R}^3 \to \mathbb{R}^2$ such that a point $q^{\mathcal{W}}$ projects to the camera point $r = \boldsymbol{H} \circ h_{\mathcal{W}}^{\mathcal{C}}(q^{\mathcal{W}})$. Under camera motion, the differential equation relating the projected point to the camera velocity for a static point $q^{\mathcal{W}}$ is

$$\dot{r} = \mathrm{D}\boldsymbol{H}(q^{\mathcal{C}}) \cdot \left(\zeta_{\boldsymbol{u}} \cdot q^{\mathcal{C}}\right), \quad \text{for } q^{\mathcal{C}} = h_{\mathcal{W}}^{\mathcal{C}} q^{\mathcal{W}} \tag{3}$$

where D is the differential operator. Since the operation $\zeta \cdot q$ is linear for $\zeta \in \mathfrak{se}(2), q \in \mathbb{R}^3$, it can be written as a matrix–vector product $\boldsymbol{M}(q)\zeta$ leading to

$$\dot{r} = \mathrm{D}\boldsymbol{H}(q^{\mathcal{C}}) \cdot \boldsymbol{M}(q^{\mathcal{C}})\zeta_{\boldsymbol{u}} = \boldsymbol{\mathcal{L}}(q^{\mathcal{C}})\zeta_{\boldsymbol{u}} \tag{4}$$

where $\boldsymbol{\mathcal{L}} : \mathbb{R}^3 \times \mathfrak{se}(2)$ is the Image Jacobian. Given the point and projection pair $(q, r) \in \mathbb{R}^3 \times \mathbb{R}^2$, $\boldsymbol{\mathcal{L}}$ works out to be

$$\boldsymbol{\mathcal{L}}(q) = \boldsymbol{\mathcal{L}}(q, r) = \begin{bmatrix} -\frac{f}{q^3} & 0 & r^2 \\ 0 & -\frac{f}{q^3} & -r^1 \end{bmatrix} \tag{5}$$

where f is the focal length. Recall that $r = \boldsymbol{H}(q)$. Re-expressing $\boldsymbol{\mathcal{L}}$ as a function of (q, r) simplifies its written form, and exposes what information is available from the image directly $r \in \mathbb{R}^2$ and what additional information must be known to compute it: coordinate q^3 from $q^{\mathcal{C}} \in \mathbb{R}^3$ in the camera frame, which is also called depth. With a stereo camera, the depth value is triangulated. The next section will use these equations for trajectory tracking with image features.

II. Trajectory Servoing

A. System Overview

The algorithmic components and information flow of a TS system are depicted in Fig. 1, and consist of two major components. The first one, described in Section II, is a TS system for a set of world points and specified trajectories. These points are obtained from the V-SLAM system as well as tracked over time. It is capable of guiding a mobile robot along short paths. The second component, described in Section III, supervises the core TS system to confirm that it has sufficient features from the feature pool to operate. Should this quantity dip too low, the supervisor queries the V-SLAM module for additional features and the robot pose to build new feature tracks.

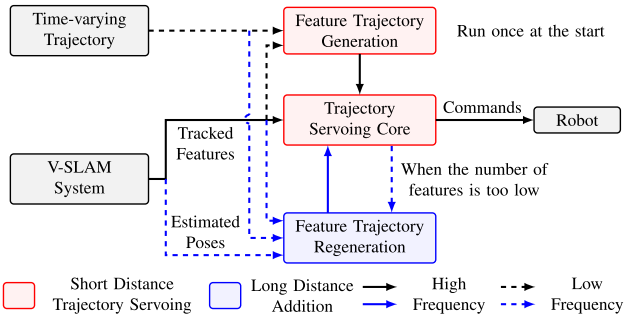


Fig. 1. TS system has two major components. One (red) steers the robot to track short paths, while the other (blue) ensures the sufficiency of features to use by querying a V-SLAM module. The entire system is used when tracking long distance trajectories. Solid arrows indicate high frequency data passing, and dashed arrows low frequency. All blue arrows represent the information flow related to long distance addition.

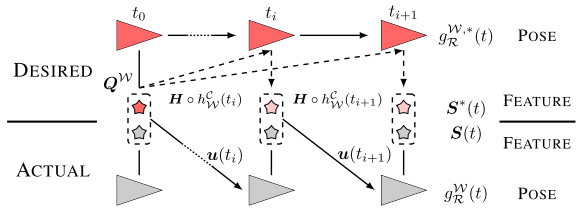


Fig. 2. TS process uses matches from $\boldsymbol{S}^*$ to $\boldsymbol{S}$ to define the control $\boldsymbol{u}$, where $\boldsymbol{S}^*(t)$ is computed from the desired trajectory $g_{\mathcal{R}}^{\mathcal{W},*}(t)$.

B. Trajectory Servoing

This section describes the basic *TS* implementation. Traditional controllers are designed in Cartesian space where V-SLAM is a pose observer (also called visual odometry). For short distance navigation, where sufficient image features remain within the field-of-view (FoV) over the trajectory, we shift the problem to image space and solve using IBVS by synthesizing a desired feature trajectory that defines what the camera should see over time from an initial view.

The standard IBVS equations presented in Section I-C typically apply to tracked features with known static positions in the world (relative to some frame attached to these positions). As described in Sections I-A1 and I-A2, a visual map or visible targets are usually necessary. The prerequisites needed for stable tracking and depth recovery of features are major challenges regarding the use of VS in unknown environments. Fortunately, they are all realizable based on information and modules available within mobile robot autonomy stacks.

TS requirements condense down to the following: 1) A set of image points, $\boldsymbol{S}^*(t_0)$, with known (relative) positions; 2) A given trajectory and control signal for the robot starting at the robot's current pose or nearby, $g_{\mathcal{R}}^{\mathcal{W}}(t_0)$; and 3) The ability to index and associate the image points across future image measurements, $\boldsymbol{S}^*(t) \leftrightarrow \boldsymbol{S}(t)$, when tracking the trajectory. In other words, it requires a mechanism to temporally associate measured features along the entire trajectory. The TS process and variables are depicted in Fig. 2. The autonomy modules contributing this information are the navigation and V-SLAM stacks. The navigation stack generates a trajectory to follow. An

indirect, feature-based V-SLAM stack keeps track of points in the local environment, links them to previously observed visual features, and estimates their actual positions relative to the robot.

1) Trajectory and Control Signals: Define $\boldsymbol{S} = \{ r_i \}_1^{n_\mathrm{F}} \subset \mathbb{R}^2$ as a set of image points in the current camera image, sourced from the set $\boldsymbol{Q} = \{ q_i^{\mathcal{W}} \}_1^{n_\mathrm{F}} \subset \mathbb{R}^3$ of points in the world frame. Suppose that the robot should attain a future pose given by g^*, for which the points in $\boldsymbol{Q}$ will project to the image coordinates $\boldsymbol{S}^* = \boldsymbol{H} \circ (g^* h_{\mathcal{C}}^{\mathcal{R}})^{-1}(\boldsymbol{Q})$. For simplicity, ignore field of view issues and occlusions between points. Their effect would be such that only a subset of the points in $\boldsymbol{Q}$ would contribute to VS.

Assume that a specific short-duration path has been established as the one to follow, and has been converted into a path relative to the robot's local frame. It either contains the current robot pose in it, or has a nearby pose. Contemporary navigation stacks have a means to synthesize both a time-varying trajectory and an associated control signal from the paths. Here, we apply a standard trajectory tracking controller [41] to generate $\xi_{\boldsymbol{u}}^*(t)$ and $g_{\mathcal{R}}^{\mathcal{W},*}(t)$ by forward simulating (1); note that $\xi_{\boldsymbol{u}}^*$ contains the linear velocity ν^* and angular velocity ω^*. Some navigation stacks use optimal control synthesis to build the trajectory. Either way, the generated trajectory is achievable by the robot.

In the time-varying trajectory tracking case, we assume that a trajectory reference $h_{\mathcal{C}}^{\mathcal{W}}(t)$ exists along with a control signal $\boldsymbol{u}^*(t)$ satisfying (2). It would typically be derived from a robot trajectory reference $g_{\mathcal{R}}^{\mathcal{W}}(t)$ and control signal $\boldsymbol{u}^*(t)$ satisfying (1). Using those time-varying functions, the equations in (2) are solved to obtain the image coordinate trajectories. Written in short-hand to expose only the main variables, the forward integrated feature trajectory $\boldsymbol{S}^*$ is

$$\begin{aligned} \dot{\boldsymbol{S}}^* &= \boldsymbol{\mathcal{L}} \circ h_{\mathcal{W}}^{\mathcal{C}}(t)(\boldsymbol{Q}^{\mathcal{W}}) \cdot \zeta_{\boldsymbol{u}^*(t)}, \text{ with} \\ \boldsymbol{S}^*(0) &= \boldsymbol{H} \circ h_{\mathcal{W}}^{\mathcal{C}}(0)(\boldsymbol{Q}^{\mathcal{W}}). \end{aligned} \tag{6}$$

It will lead to a realizable VS problem, where ν^*, ω^*, and $\boldsymbol{S}^*(t)$ are consistent with each other. The equations will require converting the reference robot trajectory to a camera trajectory $h_{\mathcal{C}}^{\mathcal{W},*}(t)$ using $\mathrm{Ad}_{h_{\mathcal{C}}^{\mathcal{R}}}^{-1}$.

2) Features and Feature Paths: The V-SLAM module provides a pool of visible features with known relative position for the current stereo frame, plus a means to assess future visibility if desired. Taking this pool to define the feature set $\boldsymbol{S}^*(0)$ gives the final piece of information needed to forward integrate (6) and generate feature trajectories $\boldsymbol{S}^*(t)$ in the left camera frame. This process acts like a short-term teach and repeat feature trajectory planner but is *simulate* and repeat, for on-the-fly generation of the repeat data.

A less involved module could be used besides a fully realized V-SLAM system, however, doing so would require creating many of the fundamental building blocks of an indirect, feature-based V-SLAM system. Given the availability of strong performing open-source, real-time V-SLAM methods, there is little need to create a custom module. In addition, an extra benefit to tracked features through V-SLAM system is that a feature map is maintained to retrieve same reappeared features. As will be shown, this significantly improves the average lifetime of features, especially compared to a simple frame by frame tracking system without the feature map.

After the V-SLAM feature tracking process, we are already working with this feasible set whereby the indexed elements in $\boldsymbol{S}$ correspond exactly to their counterpart in $\boldsymbol{S}^*$ with the same index, i.e., the sets are *in correspondence*.

3) TS Control: Define the error to be $\boldsymbol{E} = \boldsymbol{S} - \boldsymbol{S}^*$, where elements with matched indices are subtracted. The error dynamics of the points are

$$\dot{\boldsymbol{E}} = \dot{\boldsymbol{S}} - \dot{\boldsymbol{S}}^* = \boldsymbol{\mathcal{L}}_{\boldsymbol{u}}(h_{\mathcal{W}}^{\mathcal{C}}(\boldsymbol{Q}), \boldsymbol{S}; h_{\mathcal{C}}^{\mathcal{R}}) \cdot \boldsymbol{u} - \dot{\boldsymbol{S}}^* \tag{7}$$

where we apply the same argument adjustment as in (5), so that dependence is on image features then point coordinates as needed. Further, functions or operations applied to indexed sets will return an indexed set whose elements correspond to the input elements from the input indexed set. Since the desired image coordinates $\boldsymbol{S}^*$ are not with respect to a static goal pose but a dynamic feature trajectory, $\dot{\boldsymbol{S}}^* \neq 0$, see (6). Define $\boldsymbol{e}$, $\boldsymbol{s}$, $\boldsymbol{s}^*$, $\boldsymbol{q}$ and $\boldsymbol{L}$ to be the vectorized versions of $\boldsymbol{E}$, $\boldsymbol{S}$, $\boldsymbol{S}^*$, $\boldsymbol{Q}$ and $\boldsymbol{\mathcal{L}}$. Then

$$\dot{\boldsymbol{e}} = \boldsymbol{L}(h_{\mathcal{W}}^{\mathcal{C}}(\boldsymbol{q}), \boldsymbol{s}; h_{\mathcal{C}}^{\mathcal{R}}) \cdot \boldsymbol{u} - \dot{\boldsymbol{s}}^* \tag{8}$$

is an overdetermined set of equations for $\boldsymbol{u}$ when $n_\mathrm{F} \geq 2$. $\boldsymbol{L} = [\boldsymbol{L}^1, \boldsymbol{L}^2] \in \mathbb{R}^{2n_\mathrm{F} \times 2}$. Removing the functional dependence and breaking apart the different control contributions, the objective is to satisfy,

$$\dot{\boldsymbol{e}} = \boldsymbol{L} \cdot \boldsymbol{u} - \dot{\boldsymbol{s}}^* = \boldsymbol{L}^1 \nu + \boldsymbol{L}^2 \omega - \dot{\boldsymbol{s}}^* = -\lambda \boldsymbol{e}. \tag{9}$$

A least-squares solution establishes the angular rate feedback

$$\omega = \left(\boldsymbol{L}^2\right)^\dagger \left(-\boldsymbol{L}^1 \nu - \lambda \boldsymbol{e} + \dot{\boldsymbol{s}}^*\right) \tag{10}$$

so that

$$\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e} + \Delta \boldsymbol{e} \tag{11}$$

where $\Delta \boldsymbol{e}$ is mismatch between the true solution and the computed pseudoinverse solution. If the overdetermined linear system equalities (9) are compatible and have a unique solution (10), then $\Delta \boldsymbol{e}$ will vanish and the robot will achieve the target pose. If $\Delta \boldsymbol{e}$ does not vanish, then there will be an error (usually some fixed point $\boldsymbol{e}_{ss} \neq 0$). For mobile robots, it is common to use the linear velocities from $\nu^*(t)$ of the given trajectory [9], [10] for angular control (10). The decoupling provides robustness to the motion imperceptibility problem that can affect translational motion control [4], [42].

The vectorized form of (6) for $\boldsymbol{S}^*(t)$ is

$$\dot{\boldsymbol{s}}^* = \boldsymbol{L}^1(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\nu^*(t) + \boldsymbol{L}^2(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\omega^*(t). \tag{12}$$

Continuing, the vectorized steering (10) leads to

$$\begin{aligned} \omega = \left(\boldsymbol{L}^2(\boldsymbol{q}^{\mathcal{C}}, \boldsymbol{s})\right)^\dagger \Big(& \boldsymbol{L}^1(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\nu^*(t) - \boldsymbol{L}^1(\boldsymbol{q}^{\mathcal{C}}, \boldsymbol{s})\nu \\ & + \boldsymbol{L}^2(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\omega^*(t) - \lambda \boldsymbol{e} \Big). \end{aligned} \tag{13}$$

They consist of feedforward terms derived from the desired trajectory and feedback terms to drive the error to zero. The feedforward terms should cancel out the $\dot{\boldsymbol{S}}^*$ term in (7), or

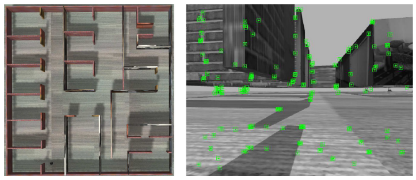


Fig. 3. Gazebo environment top view and robot view with SLAM features.

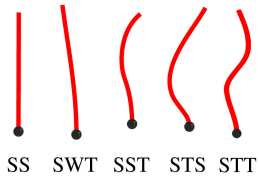


Fig. 4. Short distance template trajectories.

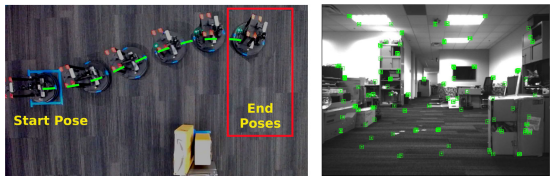


Fig. 5. Real experiment top view and robot view with SLAM features. Blue box is the robot's start pose. Red box shows the end poses region of short trajectories. The green curve is a sample trajectory to track.

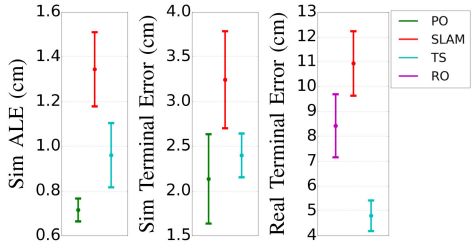


Fig. 6. 95% confidence intervals for short distance outcomes.

equivalently the now nonvanishing $\dot{s}^*$ term in (8). When traveling along the feature trajectory $\boldsymbol{S}^*(t)$, the angular velocity ω is computed from (13), where starred terms are known, and $\nu = \nu^*(t)$ from discussion after (10). As far as we know, no general IBVS tracking equations have been derived that combine feedforward and feedback signals.

C. Simulation Experiments and Results

This section runs several short distance TS experiments to evaluate the accuracy of the image-based feedback strategy supplemented by stereo V-SLAM. The hypothesis is that short distance trajectory tracking in image space will improve over tracking in pose space.

1) Experimental Setup: For quantifiable and reproducible outcomes, the ROS/Gazebo SLAM evaluation environment from [38] is used for the tests, on an Intel i7-8700 workstation. Fig. 3 shows a top–down view of the world plus a robot view. The simulated robot is a Turtlebot. It is tasked to follow a given short distance trajectory, whose desired linear velocity is 0.3 m/s and which is generated to be dynamically feasible [41]. A total of five paths were designed, loosely based on Dubins paths. They are denoted as short: straight (SS), weak turn (SWT), straight+turn (SST), turn+straight (STS), turn+turn (STT), and are depicted in Fig. 4. The average trajectory lengths are ∼4 m. Longer paths would consist of multiple short segment reflecting variations on this path set. They are designed to ensure that sufficient feature points, visible in the first frame, remain visible along the entirety of the path. Five trials per trajectory are run. The desired and actual robot poses are recorded for performance scoring.

Two metrics quantify tracking performance: 1) Average lateral error (ALE) is the two-norm of the perpendicular distance to the robot heading direction averaged over time. It measures robot deviation from the desired trajectory and has been used to evaluate steering controllers [43]; 2) Terminal error (TE) measures the robot's distance to the final stopped position of the desired trajectory after tracking ends. Implementation details are provided in the public Github repository [44].

2) SLAM Stack: Part of the robot's software stack includes the Good Feature (GF) ORB-SLAM system [30] for estimating camera poses. It is configured to work with a stereo camera and integrated into a loosely coupled, visual-inertial (VI) system [38], [45] based on a multirate filter to form a VI-SLAM system. The TS system will interface with the GF-ORB-SLAM system to have access to tracked features for servoing.

3) Methods Tested: The baseline performance standard is pose-based control using *perfect odometry* (PO) as obtained from the actual robot pose in the Gazebo simulator. PO is used for performance comparison of the tested methods. Two comparison methods are implemented. The first replaces PO with the V-SLAM estimated pose (SLAM). The second is an implementation of IBVS based on (13), which is effectively TS without the V-SLAM system. It is called VS+ to differentiate from TS, and uses a frame-by-frame stereo feature tracking system [46]. Pose-based trajectory tracking [38] uses a geometric controller with feedforward $[\nu^*, \omega^*]^T$ and feedback signals

$$\begin{aligned} \nu_{\text{cmd}} &= \nu^* \\ w_{\text{cmd}} &= k_\theta * \widetilde{\theta} + k_y * \widetilde{y} + \omega^*. \end{aligned} \tag{14}$$

where feedback uses only $\widetilde{y}$ and $\widetilde{\theta}$ from the pose error

$$[\widetilde{x}, \widetilde{y}, \widetilde{\theta}]^T \simeq \widetilde{g} = g^{-1} g^* = \left(g_{\mathcal{R}}^{\mathcal{W}}\right)^{-1}(t) \left(g_{\mathcal{R}}^{\mathcal{W},*}\right)(t). \tag{15}$$

For experimental consistency with TS, the pose-based control directly uses feedforward linear velocity terms from the given trajectory and only regulates heading. The controller gains were empirically tuned to give good performance for the PO case and extensively used in prior work [1], [2], [38]. The TS gains were also tuned [44].

4) Results and Analysis: Table I(a,b) quantify the outcomes of all methods tested. Fig. 6 consists of 95% confidence intervals of ALE and TE for the different template trajectories and methods (minus VS+) in simulations and real experiments. The first outcome to note is that VS+ fails for all paths. The average length of successful servoing is 0.4 m (∼10% of the path length).

TABLE I
SHORT DISTANCE TRAJECTORY BENCHMARK AND REAL EXPERIMENT RESULTS

(a) Sim ALE (cm)

Seq.	PO	SLAM	TS	VS+
SS	0.70	0.84	0.88	x
SWT	0.71	1.35	1.23	x
SST	0.55	1.64	0.82	x
STS	0.86	1.68	0.91	x
STT	0.75	1.21	0.96	x
Avg.	0.71	1.34	**0.96**	x

(b) Sim Terminal Error (cm)

Seq.	PO	SLAM	TS
SS	1.06	1.53	2.05
SWT	1.24	2.23	2.87
SST	2.07	3.33	2.47
STS	1.97	4.14	2.18
STT	4.34	4.99	2.42
Avg.	2.14	3.24	**2.40**

(c) Real Terminal Error (cm)

Seq.	RO	SLAM	TS
SS	4.5	8.9	4.9
SWT	6.8	10.4	4.9
SST	8.0	13.1	5.7
STS	13.3	11.9	5.0
STT	9.5	10.5	3.5
Avg.	8.4	10.9	**4.8**

Bold numbers are the average results of TS to highlight the lower gap than SLAM.

Inconsistent data association rapidly degrades the feature pool and prevents consistent use of features for servoing feedback. Without the feature map in V-SLAM, redetected features are treated as new and assigned unique indices, which violates the *correspondence* rule from Section II-B2; as noted, any effort to improve this would increasingly approach the computations found in V-SLAM. Maintaining stable feature tracking through V-SLAM is critical to TS.

Using PO as the standard, the tables show a smaller gap for TS than for SLAM as seen by the lower metric scores. For ALE, TS experiences a 35.2% degradation versus PO, while SLAM experiences an 88% degradation. The ALE statistics in Fig. 6 indicate that TS is expected to outperform SLAM. Comparing PO to TS and to SLAM, the p-values are 8e-4 and 1e-5. For TS to SLAM they are 1.9e-4. All indicate statistically significant performance differences. For TE, similar results hold except that there is overlap in the PO and TS confidence intervals. Consequently the p-value comparing PO and TS is 0.18, which is not significant. Thus, TS performs close to PO with regards to achieved terminal error, while SLAM does not (p-value: 2e-3).

TS uses the same control effort as pose-based control; the angular control efforts of the methods were not significantly different [44]. However, the rate of change of the control does differ. The time differentiated control signal norm is an indicator of control smoothness (larger means less smooth). They are 0.174, 0.174, and 1.300 for PO, SLAM, and TS, respectively. TS control is computed directly from the tracked features without temporal regularization applied to the applied controls. The SLAM estimation process smooths the orientation estimates, which translates to the control signal.

The first overall finding here is that TS outperforms SLAM and VS+, though it is based on both. This confirms that TS combines the advantages of them to enhance performance over both. Second, implementing a purely image-based approach to trajectory tracking through unknown environments is not only possible, but can work better than SLAM pose-based controls over short segments, in the absence of global positioning information. The results validate the system design proposed at the beginning of Section II-B. The robust feature tracking of V-SLAM prevents the loss of trajectory tracking stability seen in VS+. The V-SLAM feature map maintenance, feature culling, and feature retrieval modules contribute to robust VS.

D. Real Experiments and Results

To confirm that *TS* outcomes translate to practice, the short trajectory experiment is run on a LoCoBot equipped with a RealSense stereo camera and an Intel NUC (i5-7260 U). Fig. 5 presents a top-view of the experimental environment (left) and a robot view with SLAM features (right). Based on the environment and how long features can be tracked within it, the template trajectories are scaled down to ∼2.4 m. For pose-based control, two sources of robot pose estimates are tested: robot odometry (RO) and SLAM. RO is generated from onboard wheel encoders and an IMU. RO will have imperfect odometry due to measurement noise and uncertainty. Five trials were run for each trajectory and each tracking strategy. Only terminal error is measured. The continuous robot ground truth pose signal is unavailable.

1) Results and Analysis: Table I(c) and Fig. 6 give the outcomes of the tested methods. SLAM has the highest average terminal error. TS has the lowest TE average, being 42.9% lower than RO and 56.0% lower than SLAM (p-values less than 1e-2 for both). The outcomes are consistent with those from simulation: TS achieves better performance than pose-based tracking strategies using V-SLAM over short segments.

In one instance RO outperformed TS, the straight trajectory. RO pose estimation is more accurate for straight trajectories, such that odometry drift is comparable to the image-based motion imperceptibility effects that impact TS performance under zero desired angular velocities. This observation will be seen again in real experiments involving long distance TS in the following section.

2) Process Timing: Image to control timing for TS is ∼26 ms. The major time cost is V-SLAM feature tracking, with tracked feature control calculations taking ∼1.5 ms. V-SLAM image to pose estimate timing is ∼28 ms.

III. LONG DISTANCE TS

Short distance TS cannot extend to long trajectories due to feature impoverishment. When moving beyond the initially visible scene, a more comprehensive TS system should augment the feature pool $\boldsymbol{S}^*$ with new features. Likewise, if navigation consists of multiple short distance trajectories, then the system must have a regeneration mechanism for synthesizing entirely new desired feature tracks for the new segment. The overlapping needs for these two events inform the creation of a module for feature replenishment and trajectory extension.

A. Feature Replenishment

The number of feature correspondences n_{F} in $\boldsymbol{S}$ and $\boldsymbol{S}^*$ indicates whether TS can be performed without concern. Let the

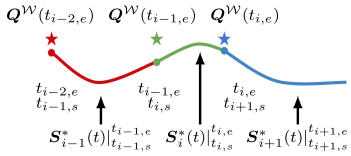


Fig. 7. Feature replenishment process. There are three segments of feature trajectories. Stars are observed point sets at corresponding time. Each circle is the start or end time of next or this segment of feature trajectory. Three feature trajectories are generated by the feature replenishment (16).

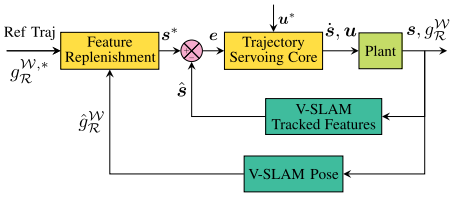


Fig. 8. Block diagram for long distance TS. Notations follow Section I and Section II. $\boldsymbol{s}^*$, $\boldsymbol{u}^*$ and $\hat{\boldsymbol{s}}$, $\hat{g}^{\mathcal{W}}_{\mathcal{R}}$ are the corresponding desired and measured values.

threshold τ_{fr} determine when feature replenishment should be triggered. Define $\boldsymbol{S}^*_i(t)|^{t_{i,e}}_{t_{i,s}}$ as the ith feature trajectory starting from $t_{i,s}$ and ending at $t_{i,e}$. The case $i = 0$ represents the first feature trajectory segment generated by (6) for $t_{i,s} = 0$, integrated up to the maximum time t_{end} of the given trajectory. The time varying function $n_{\text{F}}(t)$ is the actual number of feature correspondences between $\boldsymbol{S}(t)$ and $\boldsymbol{S}^*_i(t)$ as the robot proceeds.

When $n_{\text{F}}(t) \geq \tau_{fr}$, the feature trajectory $\boldsymbol{S}^*_i(t)$ may be used for TS. When $n_{\text{F}}(t) < \tau_{fr}$, the feature replenishment process will be triggered at the current time and noted as $t_{i,e}$. The old feature trajectory $\boldsymbol{S}^*_i(t)|^{t_{i,e}}_{t_{i,s}}$ is finished. A new feature trajectory is generated with

$$\boldsymbol{S}^*_{i+1}(t)|^{t_{i+1,e}}_{t_{i+1,s}} = \boldsymbol{H} \circ (g^*(t_{i,e}, t) h^{\mathcal{R}}_{\mathcal{C}})^{-1}(\boldsymbol{Q}^{\mathcal{W}}(t_{i,e})) \tag{16}$$

where $g^*(t_{i,e}, t)$ is the transformation between the current robot pose and a future desired pose ($t > t_{i,e}$) on the trajectory. The poses behind the robot are not included. The set $\boldsymbol{Q}^{\mathcal{W}}(t_{i,e})$ consists of observed points at the current time $t_{i,e}$. The feature pool is augmented by these current features. When this regeneration step is finished, the exact time will be assigned as the $t_{i+1,s}$. TS is performed on this new feature trajectory until the regeneration is triggered again or the arriving at the end of the trajectory. The process of regenerating new feature tracks is equivalent to dividing a long trajectory into a set of shorter segments pertaining to the generated feature trajectory segments. An depiction of feature replenishment is shown in Fig. 7.

During navigation, (16) requires the current robot pose relative to the initial pose to be known. In the absence of an absolute reference or position measurement system, the only option available is to use the estimated robot pose from V-SLAM, or some equivalent process. Although there are some drawbacks to relying on V-SLAM, it attempts to keep pose estimation as accurate as possible over long periods through feature mapping, bundle adjustment, loop closure, etc. To further couple V-SLAM and TS, we design a multiloop scheme, see Fig. 8. The inner loop is governed by TS with V-SLAM tracked features. The V-SLAM estimated pose is only explicitly used in the outer loop, during feature replenishment. Though doing so may introduce bias into the feature trajectories, the lower frequency reliance on SLAM avoids accumulating SLAM pose estimation uncertainty as would higher frequency use in the inner loop [44].

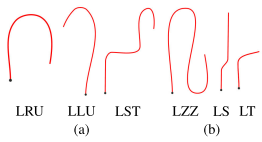


Fig. 9. Long distance trajectories. (a) Trajectories in simulation. (b) Trajectories in real experiments.

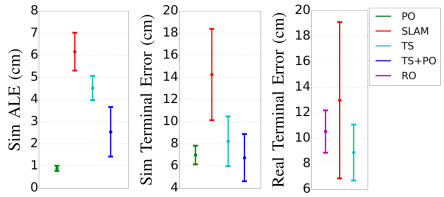


Fig. 10. 95% confidence intervals for long distance outcomes.

B. Simulation Experiments and Results

This section modifies the experiments in Section II-C to involve longer trajectories that trigger feature replenishment and synthesize new feature trajectory segments. The set of trajectories to track is depicted in Fig. 9(a). They are denoted as long: right u-turn (LRU), left u-turn (LLU), straight+turn (LST), and zig–zag (LZZ). Each trajectory is around 20 m or longer. Testing and evaluation follows as before (minus VS+). A new tracking method is added: TS+PO, which uses PO instead of SLAM odometry in the feature replenishment stage. The parameter was tuned for performance, giving $\tau_{fr} = 10$ [44].

1) Results and Analysis: Tables II(a,b) give outcomes for the two error metrics. Though TS continues to outperform SLAM and is closer to PO, the gap relative to PO is larger than the gap for short trajectories. As hypothesized, longer trajectory error is affected by the need to use SLAM pose estimates for regeneration. SLAM pose drift impacts trajectory tracking error for both methods, but is attenuated when using TS. The TS+PO outcomes confirm the impact of SLAM drift on TS performance as the TS+PO outcomes are lower than the TS outcomes. Tighter coupling of the TS and V-SLAM systems (e.g., using TS tracking to assist with pose estimation) should improve the accuracy of pose estimation, and further improve tracking performance.

Fig. 10 depicts the 95% confidence intervals of the outcomes for the template trajectories. For ALE, the intervals and p-values (<5e-3) imply that all pairwise comparisons are significant. For TE, SLAM compared to any method indicates statistical significance (p-values: <8e-3). The TE scores amongst PO, TS, and TS+PO are not significant (p-values: >0.16). TS and TS+PO performance is close to PO.

For long distance trajectories, TS again has similar control effort to pose-based feedback. However, the norms of the time differentiated control signal are 0.151, 0.148, 0.565, and 0.518 for PO, SLAM, TS, and TS+PO, respectively. TS-based control signals remain less smooth.

TABLE II
LONG DISTANCE TRAJECTORY BENCHMARK AND REAL EXPERIMENT RESULTS

(a) Sim ALE (cm)

Seq.	PO	SLAM	TS	TS+PO
LRU	0.53	3.88	4.00	1.47
LLU	0.86	8.21	5.18	1.61
LST	1.13	5.03	3.00	2.03
LZZ	1.06	7.54	5.90	5.04
Avg.	0.90	6.17	**4.52**	2.54

(b) Sim Terminal Error (cm)

Seq.	PO	SLAM	TS	TS+PO
LRU	8.57	10.66	6.42	4.15
LLU	5.54	29.48	15.75	4.02
LST	6.01	7.60	1.76	6.61
LZZ	7.83	9.28	9.00	12.21
Avg.	6.99	14.26	**8.23**	6.75

(c) Real Terminal Error (cm)

Seq.	RO	SLAM	TS
LS	8.2	14.1	10.9
LT	12.8	11.8	6.8
Avg.	10.5	13.0	**8.9**

C. Real Experiments and Results

The last experiment evaluates long trajectory performance on a real robot. The experimental setup is similar to Section II-D. Two long trajectories, LS and LT, in Fig. 9(b) are used, of lengths ∼13 m and ∼8 m, respectively.

1) Results and Analysis: In Table II(c), TS ranks first for the average TE metric. Average TE decreases 31.5% from SLAM to TS and 15.2% from RO to TS. Importantly, TS outperforms SLAM in all cases, with lower variance (see Fig. 10). These outcomes are consistent with the simulation results and support the premise behind the TS system design described at the end of Section III-A.

For the LS trajectory with long straight segments, RO outperforms SLAM because it has better pose estimation. Using TS reduces the terminal error by 22.7%, compared to SLAM. TS is more robust to pose estimation errors. Yet, it is still impacted by the SLAM pose uncertainty arising from forward motion imperceptibility due to TS' use of SLAM state information for feature regeneration. These results reproduce the observations in Section II-D1.

2) Feature Replenishment Frequency: TS feature replenishment calls queried SLAM poses 1.5 times per meter in simulation and experiment. The SLAM pose-based tracking method equivalent is 100 times per meter.

IV. CONCLUSION

This article presented an image-based trajectory tracking approach for a nonholonomic mobile ground robot. Called *TS*, it combines IBVS and V-SLAM to achieve tracking through unknown environments without externally derived absolute positioning information. TS successfully follows short trajectories. Using estimated robot poses from the V-SLAM module extends trajectory tracking to longer trajectories. Experiments demonstrate improved accuracy over pose-based trajectory tracking using estimated SLAM poses or RO. TS is less impacted by pose estimation error, by virtue of directly using for feedback the features from which pose is inferred. However, the tradeoff for relying on visual features is that feature poor environments may not be traversable using TS.

Real-world uncertainty and disturbances as well as nonsmooth trajectories may degrade tracking performance. Future work intends to improve robustness of TS by adding linear velocity control, temporal smoothness constraints, and performing uncertainty analysis on the feedback equations. In addition, more tightly coupling TS with V-SLAM may benefit the pose estimation process and further improve tracking performance.

REFERENCES

[1] J. S. Smith and P. A. Vela, "PiPS: Planning in perception space," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 6204–6209.

[2] J. S. Smith, S. Feng, F. Lyu, and P. A. Vela, *Real-Time Egocentric Navigation Using 3D Sensing*. Cham, Switzerland: Springer International Publishing, 2020, pp. 431–484.

[3] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robot. Autom. Mag.*, vol. 13, no. 4, pp. 82–90, Dec. 2006.

[4] F. Chaumette and S. Hutchinson, "Visual servo control. II. Advanced approaches," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 109–118, Mar. 2007.

[5] F. Fahimi and K. Thakur, "An alternative closed-loop vision-based control approach for unmanned aircraft systems with application to a quadrotor," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2013, pp. 353–358.

[6] D. Zheng, H. Wang, J. Wang, S. Chen, W. Chen, and X. Liang, "Image-based visual servoing of a quadrotor using virtual camera approach," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 2, pp. 972–982, Apr. 2017.

[7] D. Zheng, H. Wang, J. Wang, X. Zhang, and W. Chen, "Toward visibility guaranteed visual servoing control of quadrotor UAVs," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 3, pp. 1087–1095, Jun. 2019.

[8] A. Remazeilles, F. Chaumette, and P. Gros, "3D navigation based on a visual memory," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 2719–2725.

[9] S. Šegvić, A. Remazeilles, A. Diosi, and F. Chaumette, "A mapping and localization framework for scalable appearance-based navigation," *Comput. Vis. Image Understanding*, vol. 113, no. 2, pp. 172–187, 2009.

[10] T. Krajník, F. Majer, L. Halodová, and T. Vintr, "Navigation without localisation: Reliable teach and repeat based on the convergence theorem," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1657–1664.

[11] A. Cherubini, F. Chaumette, and G. Oriolo, "Visual servoing for path reaching with nonholonomic robots," *Robotica*, vol. 29, no. 7, pp. 1037–1048, 2011.

[12] A. Ahmadi, L. Nardi, N. Chebrolu, and C. Stachniss, "Visual servoing-based navigation for monitoring row-crop fields," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 4920–4926.

[13] A. Diosi, S. Šegvić, A. Remazeilles, and F. Chaumette, "Experimental evaluation of autonomous driving based on visual memory and image-based visual servoing," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 870–883, Sep. 2011.

[14] G. Blanc, Y. Mezouar, and P. Martinet, "Indoor navigation of a wheeled mobile robot along visual routes," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 3354–3359.

[15] L. Halodová *et al.*, "Predictive and adaptive maps for long-term visual navigation in changing environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 7033–7039.

[16] T. Do, L. C. Carrillo-Arce, and S. I. Roumeliotis, "High-speed autonomous quadrotor navigation through visual and inertial paths," *Int. J. Robot. Res.*, vol. 38, no. 4, pp. 486–504, 2019.

[17] P. Furgale and T. Barfoot, "Visual teach and repeat for long-range rover autonomy," *J. Field Robot.*, vol. 27, pp. 534–560, Sep. 2010.

[18] T. Krajník, P. Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, "Image features for visual teach-and-repeat navigation in changing environments," *Robot. Auton. Syst.*, vol. 88, pp. 127–141, 2017.

[19] N. Zhang, M. Warren, and T. D. Barfoot, "Learning place-and-time-dependent binary descriptors for long-term visual localization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 828–835.

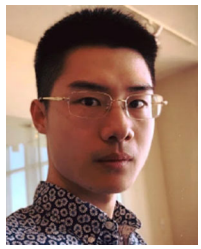
[20] T. Nguyen, G. K. Mann, R. G. Gosine, and A. Vardy, "Appearance-based visual-teach-and-repeat navigation technique for micro aerial vehicle," *J. Intell. Robot. Syst.*, vol. 84, no. 1/4, pp. 217–240, 2016.

[21] K. Kidono, J. Miura, and Y. Shirai, "Autonomous visual navigation of a mobile robot using a human-guided experience," *Robot. Auton. Syst.*, vol. 40, no. 2, pp. 121–130, 2002.

[22] A. Pfrunder, A. P. Schoellig, and T. D. Barfoot, "A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera," in *Proc. Can. Conf. Comput. Robot Vis.*, 2014, pp. 238–245.
[23] A. Vardy, "Using feature scale change for robot localization along a route," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 4830–4835.
[24] N. M. Garcia and E. Malis, "Preserving the continuity of visual servoing despite changing image features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2004, pp. 1383–1388.
[25] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.
[26] D. Fontanelli, "Mobile robot control in unknown indoor environments-the visual slam for servoing," Ph.D. dissertation, Dipartimento di Sistemi Elettrici e Automazione, Università degli Studi di Pisa, Italy, 2006. [Online]. Available: https://etd.adm.unipi.it/t/etd-05102006-093711/
[27] M. Milford and G. Wyeth, "Hybrid robot control and SLAM for persistent navigation and mapping," *Robot. Auton. Syst.*, vol. 58, no. 9, pp. 1096–1104, 2010.
[28] C. Li, X. Zhang, and H. Gao, "A general monocular visual servoing structure for mobile robots in natural scene using SLAM," in *Proc. Int. Conf. Cogn. Syst. Signal Process.*, 2018, pp. 465–476.
[29] C. Li, X. Zhang, H. Gao, R. Wang, and Y. Fang, "Bridging the gap between visual servoing and visual SLAM: A novel integrated interactive framework," *IEEE Trans. Autom. Sci. Eng.*, 2021, pp. 1–11.
[30] Y. Zhao and P. A. Vela, "Good feature matching: Toward accurate, robust VO/VSLAM with low latency," *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 657–675, Jun. 2020.
[31] L. Nardi *et al.*, "Introducing SLAMBench, A performance and accuracy benchmarking methodology for SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5783–5790.
[32] S. Saeedi *et al.*, "Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality," *Proc. IEEE*, vol. 106, no. 11, pp. 2020–2039, Nov. 2018.
[33] M. Bujanca *et al.*, "SLAMBench 3.0: Systematic automated reproducible evaluation of SLAM systems for robot vision challenges and scene understanding," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 6351–6358.
[34] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2502–2509.
[35] I. Cvišić, J. Ćesić, I. Marković, and I. Petrović, "SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles," *J. Field Robot.*, vol. 35, no. 4, pp. 578–595, 2018.
[36] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, "Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1801–1807, Jul. 2018.
[37] Y. Lin *et al.*, "Autonomous aerial navigation using monocular visual-inertial fusion," *J. Field Robot.*, vol. 35, no. 1, pp. 23–51, 2018.
[38] Y. Zhao, J. S. Smith, S. H. Karumanchi, and P. A. Vela, "Closed-loop benchmarking of stereo visual-inertial SLAM systems: Understanding the impact of drift and latency on tracking accuracy," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 1105–1112.
[39] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Automat.*, vol. 12, no. 5, pp. 651–670, Oct. 1996.
[40] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 1994.
[41] R. Olfati-Saber, "Near-identity diffeomorphisms and exponential ϵ-tracking and ϵ-stabilization of first-order nonholonomic SE(2) vehicles," in *Proc. Amer. Control Conf.*, 2002, pp. 4690–4695.
[42] R. Sharma and S. Hutchinson, "Motion perceptibility and its application to active vision-based servo control," *IEEE Trans. Robot. Automat.*, vol. 13, no. 4, pp. 607–617, Aug. 1997.
[43] S. Dominguez, A. Ali, G. Garcia, and P. Martinet, "Comparison of lateral controllers for autonomous vehicle: Experimental results," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst.*, 2016, pp. 1418–1423.
[44] S. Feng, Z. Wu, Y. Zhao, and P. Vela, "Trajectory servoing," 2022. [Online]. Available: https://github.com/ivaROS/TrajectoryServoing
[45] S. M. Weiss, "Vision based navigation for micro helicopters," Ph.D. dissertation, Dep. of Mechanical and Process Eng., ETH Zürich, Zürich, 2012. [Online]. Available: https://doi.org/10.3929/ethz-a-007344020
[46] S. Feng, "Frame-by-frame stereo feature tracking system," 2020. [Online]. Available: https://github.com/ivaROS/stereoFeatureTracking.git

Shiyu Feng (Graduate Student Member, IEEE) received the B.Eng. degree in mechanical engineering from Chongqing University, Chongqing, China, in 2015, and the M.Eng. degree in mechanical engineering from University of California, Berkeley, CA, USA, in 2016. He is currently working toward the Ph.D. degree in mechanical engineering with the School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

He is a member of Intelligent Vision and Automation Laboratory (IVALab) supervised by Dr. Patricio A. Vela. His research focuses on vision-based hierarchical navigation using sparse representation to improve the computational efficiency and scalability.



Zixuan Wu received the B.Eng. degree in automation from Harbin Institute of Technology, Harbin, China, in 2019, the M.Sc. degree from School of Electrical and Computer Engineering, Georgia Institute of Technology, USA, in 2021, and started Ph.D. program from School of Electrical and Computer Engineering, Georgia Institute of Technology, USA, in 2021.

His research interests lie in reinforcement learning, visual servoing and visual navigation. Recently, he works on the experience replay optimization for heterogeneous robot teaming.

Yipu Zhao (Member, IEEE) received the B.Sc. and M.Sc. degrees in intelligence science and technology from the Institute of Artificial Intelligence, Peking University, Beijing, China, in 2010 and 2013, respectively, and the Ph.D. degree in electrical and computer engineering from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2019 under the supervision of Patricio A. Vela, Prior to joining Meta.

He is currently a Research Scientist with Meta Reality Lab. His research interests include visual odometry/SLAM, 3D reconstruction, and multiobject tracking.

Patricio A. Vela (Member, IEEE) received the B.Sc. degree in engineering and applied sciences from the California Institute of Technology, Pasadena, CA, USA, in 1998, and the Ph.D. degree in control and dynamical systems from the California Institute of Technology, in 2003, where he did his graduate research on geometric nonlinear control and robotics.

He is currently an Associate Professor with the School of Electrical and Computer Engineering, and the Institute of Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA, USA. His research interests lie in the geometric perspectives to control theory and computer vision. Recently, he has been interested in the role that computer vision can play for achieving control-theoretic objectives of (semi-)autonomous systems. His research also covers control of nonlinear systems, typically robotic systems. In 2004, He was as a Post-Doctoral Researcher on computer vision with School of ECE, Georgia Tech. He joined the ECE faculty at Georgia Tech in 2005.