Trajectory Servoing: Image-Based Trajectory Tracking Using SLAM

Shiyu Feng\textsuperscript{1,†}, Zixuan Wu\textsuperscript{2,†}, Yipu Zhao\textsuperscript{3} and Patricio A. Vela\textsuperscript{2}

Abstract—This paper describes an image based visual servoing (IBVS) system for a nonholonomic robot to achieve good trajectory following without real-time robot pose information and without a known visual map of the environment. We call it trajectory servoing. The critical component is a feature-based, indirect 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 visual servoing. Short and long distance experiments show the benefits of trajectory servoing for navigating unknown areas without absolute positioning. Trajectory servoing is shown to be more accurate than pose-based feedback when both rely on the same underlying SLAM system.

I. INTRODUCTION

Navigation systems with real-time needs often employ hierarchical schemes that decompose navigation across multiple spatial and temporal scales. Doing so permits the navigation solution to respond in real-time 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, there are no external mechanisms to support trajectory-tracking. On-board mechanisms 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 position references. Indeed visual odometry (VO) or visual SLAM (V-SLAM) solutions are essential in these circumstances. However, they too experience drift, mostly due to the integrated effects of measurement noise. Is it possible to do better?

The hypothesis explored in this paper is that performing trajectory tracking in the visual domain can reduce the sensitivity of trajectory tracking systems reliant on VO or V-SLAM for accuracy. In essence, the trajectory tracking problem is shifted from feedback in pose space to feedback in perception space. Perception space approaches have several favorable properties when used as the local planning or trajectory synthesis module in hierarchical navigation.

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 not relevant to—or that induce sensitivities in—the local tracking task may have positive benefits. Using imaging sensors to control motion relative to visual landmarks is known as visual servoing. Thus, the objective is to explore the use of image-based visual servoing for long-distance trajectory tracking with a stereo camera as the primary sensor. The technique, which we call trajectory servoing, will be shown to have improved properties over systems reliant on VO or V-SLAM for pose-based feedback.

A. Related Work

1) 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 [1], [2]. 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 [1], [3]–[6]. 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 [3], [4]. 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 [3], [5]. In the repeat phase, the feature descriptors could be improved to make the feature matching more robust to the environment changes [6]. Visual map construction seems similar to visual SLAM. However, map construction is usually not dynamic, it is difficult to construct or update visual map in real-time while in motion because of the separation of the teach and repeat phases. In addition, VTR focuses more on the local map consistency and does not work toward global pose estimation [5] since the navigation problems it solves are usually defined in the local frame.

Another type of VTR uses the optical flow [2], [7] or feature sequence [8]–[10] along the trajectory, which is then encoded into a VTR data structure and control algorithm in the teaching phase. Although this method is similar to visual servoing, the system is largely over-determined. It can tolerate the feature tracking failure, compared with traditional visual servo system, but may lead to discontinuities [11]. Even though this method can handle relatively long trajectories since the features could always be supplemented from new teach recordings, it can only track taught trajectories.
2) **Visual Servoing:** Visual servoing (VS) has a rich history and a diverse set of strategies for stabilizing a camera to a target pose described visually. VS algorithms are classified into one of two categories: image based visual servoing (IBVS) and position based visual servoing (PBVS) [12], [13]. IBVS implementations include both feature stabilization and feature trajectory tracking [13], [14]. As a feedback strategy IBVS regulation does not guarantee what path is taken by the robot since the feature space trajectory has a nonlinear relationship with the Cartesian space trajectory of the robot. Identifying a feature path to track based on a target Cartesian space trajectory requires mapping the robot frame and the target positions into the image frame over time to generate the feature trajectory [15], precisely what is done here.

The target application is trajectory tracking for a mobile robot. Mobile robots have been studied as candidates for visual servoing [1], [2], [16]–[20]. Some of them use IBVS but do not use the full IBVS equations involving the image Jacobian. The centroid of the features [1] [19], the most frequent horizontal displacement of the matched feature pairs [2] or other qualitative cost functions [18] are used to generate [1] or correct [2] the feedforward angular velocity of mobile robot. These simplifications are reasonable since the lateral displacement of the features reflects task relevant movement that should be regulated by the robot. However, these approaches are best suited to circumstances with higher inaccuracy tolerance such as an outdoor, open field navigation. We use more precise velocity relations between the robot and feature motion to generate a feedback control signal for exact tracking similar to [16], which was experimentally verified. That work studied the path reaching problem with a visible path, which does not hold here.

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 [21]. For some autonomous robots, the SLAM pose estimates provide good signals for using pose-based feedback to track trajectories using standard control policies. The problem associated to reliance on SLAM is the potential to accrue large estimation drift, which degrades trajectory tracking and goal attainment.

Pose estimation accuracy of SLAM system is a major area of study [22]–[26]. However, most studies only test under open-loop conditions [22]–[24], [26], i.e., they only 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 the feedback control and navigation system are conducted, on individual SLAM system [27]–[29] or across different systems [30]. Closed-loop studies expose the sensitivity of pose-based feedback control and navigation, e.g., sensitivity to V-SLAM estimation drift and latency. Our work builds upon an existing closed-loop benchmarking framework and shows sensitivity reduction over conventional pose-based closed-loop navigation solution. The proposed trajectory servoing (TS) method is not tied to a specific V-SLAM implementation; the sensitivity reduction is general and applicable to other navigation systems.

### B. Contributions

The system described in this paper blends visual servoing, SLAM, and basic concepts from VTR to enable trajectory tracking of feasible paths by a mobile robot in unknown environments with sufficient visual texture. We call this combination of methods trajectory servoing because of the fact that the objective is to perform long-term trajectory tracking using visual servoing techniques. What enables this objective to be met is a stereo visual SLAM system [22], which serves as the glue tying the desired trajectory to the image information.

The algorithmic components and information flow of a trajectory servoing system are depicted in Fig. 1 and consist of two major components. The first one, which will be described in Fig. 1 is a standard trajectory tracking image-based visual servoing system for a set of known world points and specified trajectory. These known points are obtained from the SLAM system as well as tracked over time. It is capable of guiding a mobile robot along short paths. The second component, described in Fig. III supervises the core IBVS system and confirms that it always has sufficient features from the feature pool to operate. Should this quantity dip too low, it queries the SLAM module for additional features and builds new feature tracks.

Sections II and III include benchmarking and evaluation experiments quantifying the performance of the core trajectory servoing system for short paths and the entire system for long paths. Trajectory servoing is pretty unique in that short-term trajectories can be tracked as well as, or better than, pose-based feedback control with access to an oracle. Long-term accuracy is undermined by the reliance on SLAM, though it appears that trajectory servoing minimizes the reliance and exhibits more insensitivity to estimation error than pose feedback methods.

### II. Trajectory Servoing Basics

The core algorithm builds on image-based visual servoing strategies, thus this section first covers IBVS as related to the trajectory tracking problem. It then describes how IBVS is integrated into the trajectory tracking component of a standard navigation pipeline with a VO or V-SLAM component. Simple short distance trajectory tracking experiments
quantify the performance improvements that result from a short circuited closed-loop pathway for trajectory tracking.

A. Image-Based Visual Servoing

Image based visual servoing (IBVS) is most often associated to regulation or trajectory tracking for manipulators [31]. IBVS directly relates the velocity of image features to the robot velocities using image Jacobian [12], [13]. We apply this idea to non-holonomic robot motion.

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

\[
g^w_\mathcal{R} = g^w_\mathcal{R} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \nu \\ \omega \end{bmatrix} = g^w_\mathcal{R} \cdot \xi_u, \tag{1}
\]

for a forward linear velocity and \( \omega \) an angular velocity, and \( \xi_u \in se(2) \). 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 \( h^c_\mathcal{W} \) relative to the robot frame. Consequently, camera kinematics relative to the world frame are

\[
h^c_\mathcal{W} = g^c_\mathcal{W} \cdot h^c_\mathcal{R} \cdot Ad^{-1} h^c_\mathcal{R} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \nu \\ \omega \end{bmatrix} = g^c_\mathcal{W} \cdot \xi_u, \tag{2}
\]

with \( \xi_u \in se(2) \). Now, let the camera projection equations be given by the function \( H : \mathbb{R}^3 \rightarrow \mathbb{R}^2 \) such that a point \( q^w \) projects to the camera point \( r = H(q^w) \). Under camera motion, the differential equation relating the projected point to the camera velocity is

\[
\dot{r} = DH(q^c) \cdot (\xi_u \cdot q^c), \tag{3}
\]

where the point \( q \) is presumed to be static, i.e., \( \dot{q} = 0 \), and \( q^c = h^c_\mathcal{W} \cdot q^w \). Since the operation \( \xi \cdot q \) is linear for \( \xi \in se(2) \), \( q \in \mathbb{R}^3 \), it can be written as a matrix-vector product \( M(q)\xi \) leading to,

\[
\dot{r} = DH(q^c) \cdot M(q^c)\xi_u = L(q^c)\xi_u, \tag{4}
\]

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

\[
L(q) = L(q,r) = \begin{bmatrix} -\frac{r}{f} & 0 & r^2 \\ 0 & -\frac{f}{r} & -r^1 \end{bmatrix}, \tag{5}
\]

where \( f \) is the focal length. Recall that \( r = H(q) \). Re-expressing it 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 \( q^c \in \mathbb{R}^3 \) in the camera frame.

2) Image-Based Visual Servoing Equations:
Since visual servoing focuses on the control \( u \), exposing its actual structure from (2), (4) and (5), leads to

\[
\dot{r} = L(q^c) \cdot Ad^{-1} h^c_\mathcal{R} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu \\ \omega \end{bmatrix} = L_u(q^c; h^c_\mathcal{R}) \begin{bmatrix} \nu \\ \omega \end{bmatrix} \tag{6}
\]

where \( L_u \) absorbs the linear operations (the Adjoint and matrix product). IBVS then relates necessary control effort to corrective maneuvering that drives projected points in the image to target coordinates, under the assumption that the geometry of the points permits simultaneous projection to their targeted coordinates.

Define \( S = \{ r_i \}_{i=1}^n \subset \mathbb{R}^2 \) as a set of image points in the current camera image, sourced from the set \( Q = \{ q^w_i \}_{i=1}^n \subset \mathbb{R}^3 \) of points in the world frame. Suppose that the robot should attain a future pose given by \( g^* \), for which the points in \( Q \) will project to the image coordinates \( S^* = H(q^* h^c_\mathcal{R})^{-1}(Q) \). For simplicity, ignore field of view issues and occlusions between points. Their effect would be such that only a subset of the points in \( Q \) would permit use for visual servoing. Assume that we are already working with this feasible set whereby the indexed elements in \( S \) correspond exactly to their counterpart in \( S^* \) with the same index, i.e., the sets are in correspondence.

Define the error to be \( E = S - S^* \) where elements with matched indices are subtracted. The error dynamics of the points are:

\[
E = \dot{S} - \dot{S}^* = L_u(h^c_\mathcal{W}(Q), S; h^c_\mathcal{R}) \cdot u - \dot{S}^* \tag{7}
\]

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 \( S^* \) are with respect to a static goal pose, \( S^* = 0 \). Define, \( e, s, s^* \), and \( L \) to be the vectorized versions of \( E, S, S^* \) and \( L \). Then,

\[
\dot{e} = L(h^c_\mathcal{W}(q), s; h^c_\mathcal{R}) \cdot u - s^* = L(h^c_\mathcal{W}(q), s; h^c_\mathcal{R}) \cdot u \tag{8}
\]

is an overdetermined set of equations for \( u \) when \( n_r > 2 \). Removing the functional dependence and breaking apart the different control contributions, the objective is to satisfy,

\[
\dot{e} = L \cdot u = L^1 \nu + L^2 \omega = -\lambda e. \tag{9}
\]

A solution is to define,

\[
\omega = (L^2)^+ \left( -L^1 \nu - \lambda e \right), \tag{10}
\]

so that

\[
\dot{e} = -\lambda e + \Delta e, \tag{11}
\]

where \( \Delta e \) is mismatch between the true solution and the computed pseudo-inverse solution. If the problem is realizable, then \( \Delta e \) will vanish and the robot will achieve the target pose. If \( \Delta e \) does not vanish, then there will be an error (usually some fixed point \( e_{ss} \neq 0 \)). It is common for the robot’s forward velocity to be set to a reasonable constant \( \nu = \nu \) in the angular control solution (9). This law drives the camera frame to the target pose (relative to \( \mathcal{W} \)).

In the time-varying, trajectory tracking case we assume that a trajectory reference \( h^c_\mathcal{W}(t) \) exists along with a control signal \( u^*(t) \) satisfying (2). It would typically be derived from a robot trajectory reference \( g^w_\mathcal{R}(t) \) and control signal \( u^*(t) \) satisfying (1). Using those time-varying functions, the
The IBVS equations presented in the previous section are for tracking a specified camera trajectory using image feature point measurements with known static positions in the world. The prerequisites needed to solve \((12)\) and implement \((13)\) are the major challenges regarding the use of visual servoing to navigate unknown environments. They are all possible to meet based on the information and modules available within the software stack of an autonomous mobile robot. This section describes the basic trajectory servoing implementation and describes how to build a solution that satisfies the prerequisites. It focuses on short-term navigation where sufficient image features remain within the field of view for the entire trajectory.

The trajectory servoing needs to condense down to the following: (1) A set of points with known (relative) positions; (2) A given trajectory and control signal for the robot starting at the robot’s current pose (or nearby); and (3) The ability to index and associate them across future image measurements when tracking the trajectory. The autonomy modules that contribute this information are the navigation and SLAM stacks. A typical hierarchical navigation stack will generate a local trajectory to follow in realization of a long-term objective. An indirect, feature-based SLAM stack will keep track of points in the local environment and link them to previously observed visual features while also estimating their actual position relative to the robot.

1) The trajectory and control signals: 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 to generate \(\xi_n^*(t)\) and \(h_{Wt}^W(t)\) by forward simulating \((1)\) (note that \(\xi_n^*\) contains the linear velocity \(\nu^*\) and angular velocity \(\omega^*\)). Some navigation stacks use optimal control synthesis to build the trajectory. Either way, the generated trajectory is achievable by the robot, which means that the forward integrated terms in \((12)\) will lead to a realizable visual servoing problem where \(\nu^*\), \(\omega^*\), and \(s^*(t)\) are consistent with each other. The equations will require converting the robot trajectory to a camera trajectory \(h_{Ct}^W(t)\) using \(A_{h_{Wt}^W}h_{h_{Ct}^W}^{-1}\).

2) The features and feature paths: The 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 \(S^*(0)\) gives the final piece of information needed to forward integrate \((12)\) and generate feature trajectories \(S^*(t)\) in the left camera frame. This process acts like a short-term teach and repeat feature trajectory planner but is sim and repeat.

A less involved module could be used besides a fully realized SLAM system, however doing so would require creating many of the fundamental building blocks of an indirect, feature-based SLAM system. Given the availability of strong performing open-source, real-time SLAM methods, there is little need to create a custom module.

3) Trajectory Servoing Control: When traveling along the feature trajectory \(S^*(t)\), the angular velocity \(\omega\) is computed from \((13)\), where starred terms and \(\nu\) are known quantities.

C. Experiments and Results

This section runs several short distance trajectory servoing experiments to evaluate the accuracy of the image-based feedback strategy supplemented by stereo SLAM. The hypothesis is that mapping trajectory tracking to image-space will improve short term trajectory tracking by removing the impact of SLAM estimation drift from the feedback loop.

1) Experimental Setup: For quantifiable and reproducible outcomes, the ROS/Gazebo SLAM evaluation environment from [30] is used for the tests. Fig. 2 shows a top down view of the world plus a robot view. The simulated robot is a Turtlebot. In addition to the trajectory servoing algorithm, pose feedback using SLAM and using ground truth (GT) odometry will serve as comparison implementations. The controller is a linear trajectory tracking controller with feedforward and feedback terms whose gains are tuned to give good performance for the GT case.

The robot is tasked to follow a specific short-distance trajectory. A total of five paths were designed, loosely based on Dubins paths. The are denoted as straight (S), weak turn (WT), straight+turn (ST), turn+straight (TS), turn+turn (TT), and are depicted in Fig. 3. 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. Performance metrics computed are average translation error (ATE).

2) SLAM Stack: Part of the robot’s software stack includes the Good Feature (GF) ORB-SLAM system [22] for estimating camera poses. It is configured to work with a
stereo camera and integrated into a loosely coupled, visual-inertial (VI) system [30], [32] based on a multi-rate filter to form a VI-SLAM system. The trajectory servoing system will interface with the GF-ORB-SLAM system to have access to tracked features for IBVS.

3) Results and Analysis: Fig. 4 consists of boxplots of the trajectory tracking error for the different template trajectories. The trajectory servoing system has the lowest error of the methods, which demonstrates that the hypothesis was met. Interestingly, it also outperforms the case of feedback using the true pose. The result effectively show that implementing a purely image-based approach to trajectory tracking using the true pose is possible, and can work well over short segments in the absence of global positioning information. The same statistics are given in Table II and show TS outperforming GT and SLAM by 36% and 61%.

Looking at the estimation ATE, the TS method results in 14% lower translational drift indicating that employing trajectory servoing as the feedback strategy may assist SLAM. Note that the SLAM system is decoupled from the trajectory tracking system in that ORB-SLAM uses a motion prior without concern. Let the threshold \( \tau_{fr} \) determine when feature replenishment should be triggered. Define \( S^*_t(t) \) as the new feature trajectory starting from \( i,s \) and ending at \( i,e \). The case \( i = 0 \) represents the first feature trajectory segment generated by (12) for \( i,s = 0 \), integrated up to the maximum time \( t_{end} \) of the given trajectory. The time varying function \( n_F(t) \) is the actual number of feature correspondences between \( S(t) \) and \( S^*_t(t) \) as the robot proceeds.

When \( n_F(t) \geq \tau_{fr} \), the trajectory \( S^*_t(t) \) may be used for trajectory servoing. When \( n_F(t) < \tau_{fr} \), the feature replenishment process will be triggered at the current time and noted as \( t_{i,e} \). The old feature trajectory \( S^*_t(t) \) is finished. A new feature trajectory is generated with

\[
S^*_{t+1}(t)\big|_{t_{i+1,s}} = H \circ (g^*(t_{i+1,e},t) h^*_{i+1,s} W(t_{i,e}))
\]

where \( g^*(t_{i,e},t) \) is the transformation between 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 \( Q^W(t_{i,e}) \) 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 \( t_{i+1,s} \). Trajectory servoing is performed on this new feature trajectory until the regeneration is triggered beyond the initially visible scene, a more comprehensive trajectory servoing system would augment the feature pool \( S^* \) with new features. Likewise, if navigation consists of multiple short distance trajectories, then the system must have a regeneration mechanism for synthesizing an entirely new desired feature tracks for the new segment. The overlapping needs for these two events informs the creation of a module for feature replenishment and trajectory extension.

### A. Feature Replenishment

The number of feature correspondences \( n_F \) in \( S \) and \( S^* \) indicates whether trajectory servoing can be performed without concern. Let the threshold \( \tau_{fr} \) determine when feature replenishment should be triggered. Define \( S^*_t(t) \) as the new feature trajectory starting from \( i,s \) and ending at \( i,e \). The case \( i = 0 \) represents the first feature trajectory segment generated by (12) for \( i,s = 0 \), integrated up to the maximum time \( t_{end} \) of the given trajectory. The time varying function \( n_F(t) \) is the actual number of feature correspondences between \( S(t) \) and \( S^*_t(t) \) as the robot proceeds.

When \( n_F(t) \geq \tau_{fr} \), the trajectory \( S^*_t(t) \) may be used for trajectory servoing. When \( n_F(t) < \tau_{fr} \), the feature replenishment process will be triggered at the current time and noted as \( t_{i,e} \). The old feature trajectory \( S^*_t(t) \) is finished. A new feature trajectory is generated with

\[
S^*_{t+1}(t)\big|_{t_{i+1,s}} = H \circ (g^*(t_{i+1,e},t) h^*_{i+1,s} W(t_{i,e}))
\]

where \( g^*(t_{i,e},t) \) is the transformation between 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 \( Q^W(t_{i,e}) \) 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 \( t_{i+1,s} \). Trajectory servoing is performed on this new feature trajectory until the regeneration is triggered beyond the initially visible scene, a more comprehensive trajectory servoing system would augment the feature pool \( S^* \) with new features. Likewise, if navigation consists of multiple short distance trajectories, then the system must have a regeneration mechanism for synthesizing an entirely new desired feature tracks for the new segment. The overlapping needs for these two events informs the creation of a module for feature replenishment and trajectory extension.

### III. Long Distance Trajectory Servoing

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

### TABLE I

| Seq. | GT | SLAM | TS |
|------|----|------|----|
| S    | 0.0223 | 0.0361 | 0.0163 |
| T    | 0.0142 | 0.0338 | 0.0106 |
| ST   | 0.0279 | 0.0313 | 0.0151 |
| TT   | 0.0183 | 0.0354 | 0.0115 |
| Avg. ATE | 0.0209 | 0.0339 | 0.0132 |

### TABLE II

| Seq. | SLAM | TS |
|------|------|----|
| S    | 0.0116 | 0.0086 |
| T    | 0.0163 | 0.0154 |
| ST   | 0.0248 | 0.0256 |
| TS   | 0.0422 | 0.0355 |
| TT   | 0.0375 | 0.0287 |
| Avg. ATE | 0.0265 | 0.0228 |

### TABLE III

| Seq. | GT | SLAM | TS |
|------|----|------|----|
| S    | 1.5185 | 0.0386 | 1.9771 | 0.0500 | 1.5147 | 0.2534 |
| T    | 1.1260 | 0.0775 | 1.4660 | 0.0971 | 1.1254 | 0.2230 |
| ST   | 0.9497 | 0.4573 | 1.8794 | 0.8972 | 0.8478 | 0.4215 |
| TS   | 0.8029 | 0.4970 | 1.0315 | 0.6236 | 0.8038 | 0.5468 |
| TT   | 0.7488 | 0.4262 | 0.9768 | 0.5409 | 0.7485 | 0.5210 |
| Avg. | 1.0292 | 0.2993 | 1.4644 | 0.4418 | 1.0080 | 0.3931 |

Fig. 2. Gazebo environment top view and robot view with SLAM features.

Fig. 3. Short-distance template trajectories.

Fig. 4. Short-distance trajectory benchmarking results.
TABLE IV

| Seq. | GT   | SLAM | TS   |
|------|------|------|------|
| RU   | 0.0292 | 0.0627 | 0.0565 |
| LU   | 0.0354 | 0.1755 | 0.1142 |
| ST   | 0.0230 | 0.0790 | 0.0763 |
| ZZ   | 0.0469 | 0.1140 | 0.1088 |
| Avg. ATE | 0.0336 | 0.1078 | 0.0890 |

TABLE V

| Seq. | SLAM | Traj. Servo |
|------|------|-------------|
| RU   | 0.0917 | 0.0896 |
| LU   | 0.1387 | 0.1434 |
| ST   | 0.0702 | 0.0691 |
| ZZ   | 0.1212 | 0.1188 |
| Avg. ATE | 0.1055 | 0.1052 |

TABLE VI

| Seq. | GT | SLAM | Traj. Servo |
|------|----|------|-------------|
| RU   | 3.0871 | 2.6438 | 2.0536 |
| LU   | 2.5200 | 1.3544 | 1.7756 |
| ST   | 2.0466 | 1.5580 | 2.0356 |
| ZZ   | 2.3257 | 1.1764 | 2.3279 |
| Avg. | 2.4949 | 1.6832 | 2.4990 |

Fig. 5. Long-distance trajectories

Fig. 6. Long trajectory benchmarking results

B. Experiments and Results

This section modifies the experiments in §II-C to involve longer trajectories that will trigger feature replenishment and synthesize new feature trajectory segments. The set of trajectories to track is depicted in Fig. 5. They are denoted as right u-turn (RU), left u-turn (LU), straight+turn (ST), and zig-zag (ZZ). Testing and evaluation follows as before.

1) Results and Analysis: Fig. 5 consists of boxplots of the translation-only trajectory tracking ATE for the template trajectories. As hypothesized, the error over longer trajectories is affected by the need to use SLAM pose estimates for regeneration. However, it appears that in the one case where the SLAM system performed the worst, the TS approach had better performance. The tabulated results in Tab. IV indicates that on average the TS approach improved over SLAM by 17% in terms of tracking error. What is interesting is that the position estimation ATE of both systems is comparable, which indicates that trajectory servoing may have better closed-loop noise rejection properties by working directly in image-space instead of using inferred pose estimates from the SLAM estimator. Table VI suggests that there may be an increase in steering control effort for long term trajectory servoing.

C. Navigation with online path planning

In addition, in order to show how trajectory servoing works in the real navigation task. We move the environment down to the ground and create a 2D occupancy map. Without using predefined paths, given a goal, the global planner can find a collision-free path from the map. The online global path is used for the robot to apply trajectory servoing. The full system in Fig. 1 is used. Fig. 7 shows one successful navigation task. The robot starts from the blue point, and ends at 5 different positions, annotated as orange points. The average ATE for GT, SLAM and TS are 0.0125m, 0.0569m and 0.0354m. Trajectory servoing, enabled by VI-SLAM, has better performance than using VI-SLAM for pose-based feedback.

IV. Conclusion

In the paper, we present a vision-based navigation approach for non-holonomic mobile robot, so called trajectory servoing, which combines the IBVS and SLAM. This trajectory tracking method can successfully follow a given short trajectory without any robot pose information. By integrating with estimated robot poses from SLAM, it can achieve long-term navigation. The SLAM system also provides robust feature tracking to support stable trajectory servoing. The experiments demonstrate the higher tracking accuracy than pose-based trajectory tracking with estimated SLAM poses.
