State Representation Learning for Visual Servo Control

Wang, Jen-Wei; Nikovski, Daniel N.

TR2025-094 June 26, 2025

Abstract

We propose a method for visual servo-control of robots using images from an uncalibrated camera that constructs compact state representations of the robot's con- figuration and uses transition dynamics learned from collected execution traces to compute control velocities to reach a desired goal state identified directly by its image. The key step of the proposed method is the estimation of a homography transform between the image positions of distinct keypoints belonging to the robot in the current image and those in a reference image, which can be done quickly and robustly even when not the same set of keypoints is observed at each time step, making it robust to noise and variations in illumination. The estimated homography is then used to represent the robot configuration as the image coordinates of a minimal number of virtual points moving with the robot. The method was verified experimentally for planar motion of a fully actuated manipulator arm as well as an underactuated mobile robot with a nonholonomic constraint.

European Control Conference (ECC) 2025

State Representation Learning for Visual Servo Control

Jen-Wei Wang and Daniel Nikovski[†]

Abstract— **We propose a method for visual servo-control of robots using images from an uncalibrated camera that constructs compact state representations of the robot's configuration and uses transition dynamics learned from collected execution traces to compute control velocities to reach a desired goal state identified directly by its image. The key step of the proposed method is the estimation of a homography transform between the image positions of distinct keypoints belonging to the robot in the current image and those in a reference image, which can be done quickly and robustly even when not the same set of keypoints is observed at each time step, making it robust to noise and variations in illumination. The estimated homography is then used to represent the robot configuration as the image coordinates of a minimal number of virtual points moving with the robot. The method was verified experimentally for planar motion of a fully actuated manipulator arm as well as an underactuated mobile robot with a nonholonomic constraint.**

I. Introduction

Visual servo (VS) control methods use information from images acquired by cameras and extracted by means of computer vision algorithms to control the operation of a robot [1]. The use of VS methods is economically appealing when the pose of the robot and/or objects manipulated by it are unknown and not directly measurable, but can be observed by one or more cameras.

A straightforward method to use images for VS is to apply computer vision to determine the pose of the robot and/or objects in the inertial frame and then use these poses in the control loop. This method, known as pose-based VS (PBVS) has the advantage of reducing the very high-dimensional image signal to the most compact representation possible of the configuration of the objects of interest in the scene. If known, the dynamics of the robot and manipulated objects can be represented in the same task (Cartesian) space, allowing the use of model-based state-space controller design methods to achieve desired control objectives.

However, PBVS requires the use of calibrated cameras in order to recover the Cartesian poses of objects in the scene from camera images. Furthermore, often CAD models of the observed objects are needed to achieve acceptable accuracy of pose estimation. When such calibration is not available, VS can still be performed successfully by closing the control loop not on actual Cartesian poses, but on a vector of visual features $\mathbf{s}(t)$ [1]. If the values $\mathbf{s}^*$ of the features at a desired goal state are known, a VS error can be defined as $\mathbf{e}(t) = \mathbf{s}(t) - \mathbf{s}^*$ and used in a feedback VS control scheme. A central question in such VS schemes is what the feature vector $\mathbf{s}(t)$ should be. The feature vector can consist of suitable visual measurements, such as the coordinates of reliably identifiable unique keypoints in the image, such as corners, edges, color blobs, etc. This more general approach is known as image-based VS (IBVS), and is very attractive economically due to eliminating the need for camera calibration or CAD models.

[†]All authors are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA 02139 {jenwang,nikovski}@merl.com

However, using IBVS requires deciding which visual features to use, which is usually problem-dependent and involves costly human labor. Furthermore, if the feature vector does not consist of actual Cartesian poses, any available knowledge about the system's dynamics expressed in terms of these poses cannot be used to design a feedback VS controller, so different controller design methods are necessary. It is thus very desirable to automate the process of feature vector selection and VS controller design, based on collected experimental data and leveraging machine learning techniques. As the feature vector $\mathbf{s}$ is a proxy for the actual configuration of the system, the automation of this process effectively consists of state representation learning (SRL) from image data followed by model-based controller design using learned dynamics in the constructed state space representation.

Section II defines the version of the VS control problem we are addressing. Section III reviews the common methods for VS control with and without calibrated cameras and known system dynamics, as well as the complications in VS control arising from nonholonomic constraints. Section IV proposes a novel method for SRL from observed visual data during planar motion, and VS control based on learned nonlinear dynamical models in the constructed state space. Section V introduces two experimental platforms for object handling using a robotic manipulator and navigation of a mobile robot. Section VI describes experiments to verify empirically the proposed method on both experimental platforms. Section VII discusses future work and concludes.

II. Problem Definition

We are considering the VS control of a robot or object undergoing planar motion whose configuration $\mathbf{q} \in \mathcal{C}$ is described by the vector $\mathbf{q} = (x, y, \theta)^T$ in an inertial frame, where the configuration space $\mathcal{C}$ coincides with $\mathbb{R}^2 \times \mathbf{S}^1$. An example of such motion is a mobile robot moving around a flat office floor, or an object grasped by a robot arm and moved parallel to a flat work surface in a work cell. The robot is steered in velocity control mode by control inputs $\mathbf{u} = [\mathbf{v}, \omega]$, where $\mathbf{v}$ is linear velocity in the plane of motion and ω is angular velocity about an axis perpendicular to that plane. The use of velocity control mode, available on most mobile robots and industrial manipulator arms, reduces the normally second-order dynamics of motion to a first-order kinematic model with trivial time-invariant system dynamics $\dot{\mathbf{q}} = \mathbf{u}$. However, if the true configuration of the system $\mathbf{q}$ cannot be recovered from sensor measurements, state-space controller design based on these simple dynamics cannot be performed, because an observer cannot be designed for a full-state feedback controller.

Instead, we assume that camera images $I(t)$ at any instant in time contain sufficient information to identify unambiguously the configuration $\mathbf{q}(t)$. Our goal is then to construct (learn) a suitable feature vector $\mathbf{s}[k]$ from a sequence of images $I[k] = I(t_k)$ collected at discrete time instants $t_k = k\Delta t$, where Δt is the control time step, and associated image measurements, by exciting the system with control inputs $\mathbf{u}[k] = \mathbf{u}(t_k)$ applied at these instants. The transition function $\mathbf{s}[k+1] = \mathbf{f}(\mathbf{s}[k], \mathbf{u}[k])$ in the constructed feature space is no longer trivial, but we can learn it from the same execution trace, and design a VS controller based on it.

There are two common methods of placement of the camera relative to the robot: eye-in-hand (moving with the robot) and eye-to-hand (stationary) [1]. Without loss of generality, we consider the eye-to-hand placement. We assume that the image plane is not necessarily parallel to the xy plane of motion of the robot in Cartesian space, but nevertheless, in most practical settings, it is advantageous to point the camera generally close to perpendicular to the plane of motion of the robot, in order to maximize the camera's field of view.

III. Related Work on Visual Servo Control

In PBVS, because $\mathbf{s}[k] = \mathbf{q}[k]$ and $\dot{\mathbf{q}} = \mathbf{u}$, we have $\dot{\mathbf{s}} = \mathbf{u}$, and a simple proportional control scheme is $\mathbf{u}[k] = -\lambda\mathbf{e}[k]$, where $\mathbf{e}[k] = \mathbf{s}[k] - \mathbf{s}^*$, with a proportional gain λ [1]. This proportional controller will work well in systems without nonholonomic constraints, gradually reducing the visual servoing error to zero, when the robot will assume its desired configuration $\mathbf{s}^*$.

However, in systems with nonholonomic constraints, such as wheeled mobile robots, the number of controls usually does not equal the number of degrees of freedom, and the robot cannot move in a direction perpendicular to that of its wheels. For a planar mobile robot, only the magnitude of its linear velocity $v = \|\mathbf{v}\|$ can be controlled, but not its direction, as that direction is constrained to coincide with that of its wheels. Consequently, when the visual feedback error is perpendicular to the orientation of the robot's wheels, if the simple proportional feedback control scheme from above is used, the robot will not move and will never close the feedback error. For such systems, suitable motion paths must be computed by control methods for control of nonholonomic robots [2], [3].

In IBVS, $\mathbf{s}[k] \neq \mathbf{q}[k]$, but the relationship between $\dot{\mathbf{s}}$ and $\mathbf{u}$ can still be expressed by the linearization of the dynamics $\dot{\mathbf{s}} = L_s\mathbf{u}$, where the matrix L_s is called the interaction matrix related to $\mathbf{s}$. It can be the actual Jacobian of the nonlinear dynamics $\mathbf{f}(\mathbf{s}, \mathbf{u})$ around $\mathbf{s}$, but it can also be the linearization around a different point, for example the goal state $\mathbf{s}^*$, or an average of several Jacobians [1]. The relationship between the derivative of the error and control velocity can then be expressed as $\dot{\mathbf{e}} = L_s\mathbf{u}$, leading to a proportional controller of the form $\mathbf{u}[k] = -\lambda L_s^+\mathbf{e}[k]$, where $L_s^+ = (L_s^T L_s)^{-1}L_s^T$ is the Moore-Penrose pseudo-inverse of L_s, accounting for the fact that the length of the vector $\mathbf{s}$ can be different (usually longer) than that of the configuration $\mathbf{q}$ or control vector $\mathbf{u}$.

The choice of image features $\mathbf{s}$ and the method for computing the interaction matrix L_s are of primary importance in VS control schemes. In IBVS, the features are usually the image-plane normalized coordinates of trackable keypoints in the image, meaning that the ability to reliably track these points without fail is absolutely critical for success.

The exact interaction matrix L_s can be computed analytically only if the current depth of each point in the camera frame is known, which is not easy to ensure unless the image plane is always parallel and at a constant distance to a plane in which the points move. As noted in the previous section, we are not making this assumption, as it is difficult and inconvenient to enforce in practice. Some IBVS methods use pose estimation methods to compute point depths at every iteration, or choose a constant interaction matrix computed at the goal position $\mathbf{s}^*$ [1].

When depth information is not available, or the camera is not calibrated, off-line learning or on-line estimation methods can be used to directly estimate the numerical values of the interaction matrix from experimental data in the form of feature variations $\Delta\mathbf{s}$ caused by commanded pose variations $\Delta\mathbf{u} = \mathbf{u}\Delta t$. A single interaction matrix for a specific pose can be estimated by means of least-squares regression, and neural networks and other machine learning methods can be used to learn pose-dependent interaction matrices [1].

Similar to PBVS, nonholonomic constraints and/or underactuation complicate IBVS control significantly. As noted, even if the camera is calibrated and the robot's pose can be estimated in the world frame, the task of servoing the robot to a goal state defined directly in configuration space is not trivial and usually involves the application of planning algorithms. Such methods usually require an accurate kinematic model of the robot's motion [2]. Furthermore, many of these algorithms rely critically on the ability to compute the current bearing of the robot with respect to the goal. However, if the camera is not calibrated and the true pose (configuration) cannot be estimated, the heading of the robot in the world frame cannot be computed as part of its pose, so it becomes impossible to apply such algorithms.

A method proposed by Zhang et al. [4] performs visual servoing on a mobile robot equipped with an uncalibrated camera limited to the eye-in-hand setting. In their method, the unknown extrinsic camera parameter is the planar angle between the robot and camera frames, and the algorithm performs an iterative parameter identification step in order to estimate it from collected data. Although this solution is effective, it is specific to the eye-in-hand setting, and cannot be applied easily to the eye-to-hand setting. Another set of algorithms [5], [6] solves the problem assuming that the plane of the robot's motion is parallel to the image plane, which, as noted, is often impossible to guarantee.

IV. Proposed Method for State Representation Learning and Visual Servoing

We propose an IBVS method for robots equipped with pinhole uncalibrated cameras, where the feature vector is constructed automatically based on sequences of images collected in a self-supervised manner. It addresses one of the main shortcomings of most IBVS methods – the need to always reliably track specific keypoints in the image, resulting in failure of the controller if one of more of them are lost in an image frame. The method still leverages algorithms for identifying keypoints in images and establishing matching correspondences for such keypoints in pairs of images, but is robust to failure to track such points in each and every image. Instead, the method estimates a homography matrix from whatever keypoint correspondences are available for a new frame, and computes the expected image coordinates of two virtual points that jointly define a feature vector that uniquely and robustly identifies the robot's configuration. Using a sequence of images and the compact feature vectors computed in this way, the proposed algorithm learns a model of transition dynamics in feature space using standard supervised ML methods. The learned model is then used to design a feedback VS controller, depending on the type of robot controlled. For fully actuated robots without nonholonomic constraints, the learned transition model can be differentiated to obtain an interaction matrix used in a proportional VS control scheme. For robots with nonholonomic constraints, a method based on differential dynamic programming is used to compute a trajectory to the goal state in feature space, and locally linear controllers are used to track this trajectory by relating deviations from it to suitable velocity commands that will bring the robot back on track.

The algorithm learns relatively compact state descriptors of a robot moving in a plane and observed by a fixed overhead camera. The method tracks image keypoints in the camera image that correspond to coplanar points moving together with the robot. The positions of these points are unknown, but remain constant in the robot frame.

The robot moving in the plane is observed by a stationary pinhole uncalibrated camera with unknown extrinsics. It can measure the coordinates of the points $\bar{\mathbf{p}}_i[k] = (\bar{x}[k], \bar{y}[k])^T$, $i = 1, \ldots, n$ in its own camera frame (2D image coordinates) captured at time instant t_k. The fact that the observed points are coplanar means that there exists an unknown, but constant homography $H_{k,j}$ relating the image coordinates of *all* corresponding points measured at times k and j: $\lambda_{k,j}\tilde{\bar{\mathbf{p}}}_i[k] = H_{k,j}\tilde{\bar{\mathbf{p}}}_i[j]$, where the homogeneous coordinates $\tilde{\bar{\mathbf{p}}}_i = [\bar{\mathbf{p}}_i^T, 1]^T$ all have a scaling factor of 1 [7]. Note that the scaling factor $\lambda_{k,j}$ is constant for all points $i = 1, \ldots, n$.

Let $\bar{p}_i^*$ be the measured keypoint in image coordinates for the goal configuration of the robot. Given $n \geq 4$ points $\bar{p}_i[k]$ whose correspondence has been established between time instance k and the goal configuration, the homography $H_{k,*}$ can be computed using well-known algorithms [7]. (The reason at least 4 pairs of corresponding points are needed is that the homography matrix has 8 independent parameters and each pair of corresponding points provides 2 linear equations. Although, the more keypoints are used, the more accurate the estimate is.) However, recovering the transform corresponding to the planar motion of the robot in its plane of motion from $H_{k,*}$ for the purposes of PBVS is only possible if the intrinsic parameters of the camera are known [7], which we do not assume to be the case.

Instead, we propose to use purely IBVS, where the homography matrix $H_{k,*}$ is used to construct a compact representation of the robot's pose, without necessarily recovering that pose explicitly. This representation is based on the idea that two fixed points on the robot are sufficient to uniquely determine its pose if the robot undergoes planar motion. Let $p_1^* = (x_1^*, y_1^*)^T$ and $p_2^* = (x_2^*, y_2^*)^T$ be the coordinates of two such distinct points defined in the image of the goal configuration. (They might or might not be actual keypoints, and do not even have to lie within the shape of the robot in that image; all that is needed is that they be distinct.) These "virtual" keypoints define uniquely the desired configuration $\mathbf{s}^*$ of the robot as a 4-dimensional vector: for example, one of these two points p_1^* can be taken to be the position of the robot, and the angle $atan2(y_2^* - y_1^*, x_2^* - x_1^*)$ defines its orientation in screen coordinates, which has a (nonlinear) one-to-one mapping to the true orientation of the robot in the world frame.

Subsequently, when performing VS control at time instant k and after we have computed the homography $H_{k,*}$ from as many keypoints as were registered in the image captured at that time and matched to keypoints in the goal image, we can compute where the two virtual keypoints must be at that time as $\tilde{p}_i[k] = (x_i[k], y_i[k], 1)^T = H_{k,*}\tilde{p}_i^*/\lambda_{k,*}$, $i \in \{1, 2\}$, and $\lambda_{k,*}$ is the perspective scaling factor for the homography between image frame k and the target configuration's image. These coordinates unambiguously identify the configuration of the robot $\mathbf{s}[k] = (x_1[k], y_1[k], x_2[k], y_2[k])^T$ at time t_k.

Importantly, not all n keypoints must be visible at every time step k – as the estimation problem is overconstrained, any subset of the keypoints can be used for estimating the homography $H_{k,*}$, as long as its size exceeds the minimum threshold of four registered points.

Since the homography matrix $H_{k,*}$ represents the robot's relative movement between pose at time instance k and goal pose in image space, we only consider the correspondences of keypoints that are attached to the robot. To this end, an image segmentation algorithm is adopted to mask the robot's region in goal image I_g, creating a new goal image where the background is removed. In this way, most of the matching points will be on the robot and the outliers will be eliminated by the RANSAC algorithm during homography estimation.

Given state descriptors $\mathbf{s}[k]$, $k = 0, \ldots, N+1$ corresponding to an execution trace collected under the application of a control sequence $\mathbf{u}[k]$, $k = 0, \ldots, N$, we can learn the next-state function $\mathbf{s}[k+1] = \mathbf{f}(\mathbf{s}[k], \mathbf{u}[k])$ using any suitable ML method, such as Gaussian Process Regression (GPR) [8]. If a differentiable ML model is used to learn the dynamics, it can be differentiated analytically to compute the entries of the interaction matrix L_s needed for VS control.

For fully actuated holonomic systems, the interaction matrix L_s would be of size 4×3 for the proposed state representation and planar motion controls (two linear and one rotational velocities), and its Moore-Penrose pseudo-inverse L_s^+ can be used directly for greedy VS control, as described in Section III. For underactuated and/or nonholonomic systems, a path planning method can be employed to find a sequence of nominal controls and states leading from the current state to the desired goal state, using the learned transition dynamics. Possible choices for such methods include differential dynamics programming (DDP) and its more modern variant the iterative linear quadratic regulator (iLQR) [9]. These methods use the local linearization of the nonlinear transition dynamics to iteratively find a local optimum of the nominal control and state trajectories in terms of a cumulative cost that expresses a preference for reaching the goal state in minimal time and possibly with minimal control effort. These methods also produce a sequence of linear feedback controllers that stabilize the system along the computed nominal trajectory.

V. Experimental Setups

We investigated the proposed algorithm experimentally on three setups: a manipulator arm constrained to move its end effector in a plane, a simulator of a planar mobile robot, and a real two-wheeled mobile robot.

A. *Object Handling with a Robotic Manipulator*

The purpose of this testbed is to evaluate the algorithm on a holonomic system. Figure 1.a shows the hardware setup of the environment which consists of a box attached rigidly to the endtool of a MELFA RV-4FL industrial six-axis robot arm and an Oak-D Pro camera looking at the box. The task is to design a VS controller that can command the robot to move the box to a desired pose. The desired pose is determined by steering manually the robotic arm to a goal configuration and capturing a goal image of the box. Because the robot has high precision ($\pm$ 0.1 mm), we can record robot poses both in current and goal configuration and use the stock Cartesian-space controller of the robot to do the goal-reaching task directly in task space, as a baseline for comparison with the VS method. That is, we regard this environment as an evaluation platform for testing how accurate our VS controller is, using ground truth about the robot's end tool obtained from the robot joints' encoders.

We use velocity control in Cartesian space to command the robot, constraining the robot's movement in the YZ plane of the world frame attached to the robot's base. This results in a total of three DoF, including translation along the y axis, translation along the z axis, and rotation about x. The total control input $\mathbf{u}^m$ is denoted by $[v_y^m, v_z^m, \omega^m]$. Note that the rotational center is not at the center of the box and we do not know the relative pose between the box and the robot's end effector.

The control loop is synchronized with the camera's frame capture, at 60 frames per second (fps). We only use every 6-th frame for VS, resulting in a control rate of 10 Hz.

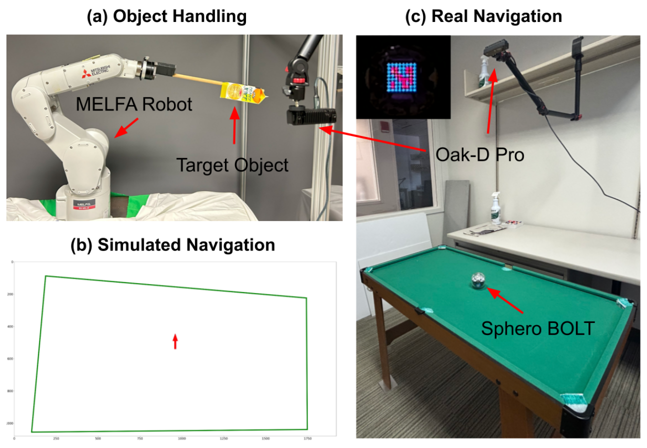

Fig. 1. Experimental testbeds in three environments. (a) A juice box attached to a MELFA RV-4FL six-axis robot. (b) A simulated navigation environment viewed from a top-down camera. Green lines show the boundary of the pool table and a red arrow represents the robot's position and heading. (c) An environment consisting of a pool table and a Sphero BOLT robot. The upper left corner shows a randomly selected LED pattern with rich features on top of the robot.

We follow the methodology from Section IV to design our tracking algorithm. We manually select two points $((x_1^*, y_1^*)^T$ and $(x_2^*, y_2^*)^T)$ on the goal image and use the estimated homography matrix to transform two points from the goal image to current image to obtain the compact representation $(x_1, y_1, x_2, y_2)^T$. Regarding homography estimation, we adopt SIFT [10] as a feature detector/descriptor to find feature points both in the current and goal images. The correspondence between two sets of feature points is found by calculating the Euclidean distance between two descriptors and the pairs with the shortest distance are selected.

B. *Navigation of a Mobile Robot*

We evaluate the algorithm on navigation tasks both in a simulated and a real-world environment. The reason for using a simulated environment is that our selected real-world mobile robot has some uncertainty in its actuation and we would like to also evaluate the algorithm in an environment with perfect representation and dynamics.

In simulation, we use the analytical unicycle model as the ground-truth dynamics to simulate the rollout of a differential-drive robot under control commands. As shown in Figure 1.b, the simulated robot is operated on a virtual table, where its boundary (green line in the figure) limits the exploration space of the robot. To simulate perspective transformation, we attach a virtual camera above the table. The camera is tilted in order to evaluate whether controllers can still perform well under a general perspective transformation and without camera calibration. The simulation environment in Figure 1.b shows the view from the top-down camera, and accordingly the green lines do not form a perfect rectangle.

Using the methodology from Section IV, two points are defined beforehand in the goal image. To simplify the dynamics formulation, the first point $(x_1, y_1)^T$ is defined to be the centroid of the robot, and the second point $(x_2, y_2)^T$

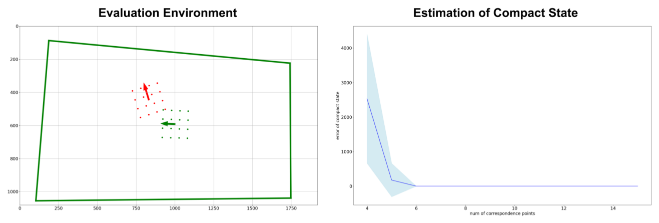

Fig. 2. Evaluation of constructed state. The green arrow represents the starting compact state and the red arrow represents the transformed compact state after a control input is applied to the robot. We assume that at least 4 pairs and at most 16 pairs of feature points can be found by the SIFT algorithm. Results show that the compact state can be robustly estimated if at least 6 pairs of feature points can be found.

is chosen to be some fixed distance away. The two points correspond to the start and end points of the red arrow in the simulation environment of Figure 1.b. Based on the given intrinsic and extrinsic matrices, the ground-truth dynamics in the image space $\mathbf{f}$ can be derived analytically to evaluate the accuracy of the proposed VS system.

In the hardware setup shown in Figure 1.c, we use a Sphero BOLT robot as our target robot and operate it on a pool table. Although the Sphero BOLT looks like an omnidirectional robot, its internal mechanism makes it similar to a unicycle robot, because it cannot move sideways without turning its heading first. A Sphero BOLT can be controlled via a commanded rotation angle θ^n and translation velocity v^n. Because velocity control is used in VS, the rotation command to the robot is obtained by multiplying the desired rotational velocity ω^n by the control step Δt.

We mounted a Luxonis Oak-D Pro camera above the table as shown in Figure 1.c. In our VS system, we used the frame capture loop of the camera, running at 60 fps, and decimated it 10 times to obtain the VS control frequency of 6 Hz and control step $\Delta t = 0.1667$ ms. We follow the method described in Section IV to design the tracking algorithm. The Sphero BOLT has an array of 8x8 LEDs that can display color patterns, and we randomly selected a pattern with rich features, as shown in the upper left corner of Figure 1.c.

After the pattern is selected and displayed, we manually put the robot in the final desired pose and capture an image, stored as the goal image. Once at least 4 correspondences are found using the SIFT algorithm [10], a compact representation $(x_1, y_1, x_2, y_2)^T$ can be found by applying the estimated homography matrix to transform $(x_1^*, y_1^*)^T$ and $(x_2^*, y_2^*)^T$ on the goal image to the current image. To handle the reflection from the transparent shell of the Sphero BOLT, we applied a color filtering algorithm on the YCrCb color space, resulting in a robust tracking algorithm for the compact state.

VI. Experimental Verification

A. Evaluation of Constructed State

In this subsection, we present an investigation of the accuracy of the proposed compact state descriptor. Since we can only obtain the ground-truth compact state in the simulator, we validated the estimated virtual points in the simulated navigation environment. Furthermore, we investigated how different patterns influence the estimation results. To this end, we varied the number of correspondences found by the SIFT algorithm [10] to understand the relationship between the number of matched feature points and estimation accuracy.

Figure 2 shows the test environment and estimation results. The start and end of the red arrow represent the two virtual points. The green arrow is the starting compact state and the green dots are the detected feature points. After a control input is applied to the robot, the green arrow will move to the red one and so will the feature points. We randomly select 4 to 16 pairs of feature points from these pre-defined dots and estimate the homography transformation matrix. The estimated compact state is then obtained by applying the homography matrix to the green arrow. The error is calculated via Euclidean distance between the estimated compact state and the ground-truth compact state. We place randomly the robot and apply random control inputs to obtain 500 test cases.

Based on Figure 2, we can observe that the estimation becomes accurate and robust across all the test cases if the number of correspondences exceeds 6. The blue shaded area is the standard deviation of the estimation error. If 16 correspondences can be found, the average error is **0.15** pixels and its standard deviation is **0.17** pixels. Since we assume that the robot motion on the image plane is a linear transformation, most of the error comes from the distortion of the camera on a real setup. However, the least squares solution can still obtain a robust and accurate result even when some non-linearity exists in the system.

B. Learning the Dynamics Function

Because the true dynamics are unknown, learning $\mathbf{f}$ from robot exploration data is an essential component of model-based control. However, directly learning the function $\mathbf{f}$ is not sample-efficient and collecting data for learning $\mathbf{f}$ is time consuming in real-world environments. Therefore, we assume that the translation and rotation parts of the robot's dynamics are independent from each other and the dynamics function $\mathbf{f}$ can be decomposed into several parts.

1) Object Handling: In the object handling environment, we decompose the incremental dynamics function $\mathbf{f}^m_{dec}$ into three parts $\mathbf{f}^m_y$, $\mathbf{f}^m_z$, and $\mathbf{f}^m_r$:

$$\Delta^m_y[k] = \mathbf{f}^m_y(x_1[k], y_1[k], v^m_y[k]) \tag{1}$$
$$\Delta^m_z[k] = \mathbf{f}^m_z(x_1[k], y_1[k], v^m_z[k]) \tag{2}$$
$$\Delta^m_r[k] = \mathbf{f}^m_r(x_1[k], y_1[k], x_2[k], y_2[k], \omega^m[k]) \tag{3}$$

where Δ denotes the increment of state $\mathbf{s}[k+1]-\mathbf{s}[k]$ and the subscript denotes which control dimension contributes to the state increment. Therefore, after a control input $(v^m_y, v^m_z, \omega^m)^T$ is applied, the total state increment $\Delta^m_{\mathbf{s}}[k]$ expressed as $(\Delta_{x1}, \Delta_{y1}, \Delta_{x2}, \Delta_{y2})^T$ will be the summation of three components $\Delta^m_y[k] + \Delta^m_z[k] + \Delta^m_r[k]$. Since we assume that the two virtual points are not far away from each other and the distortion effect of the camera is not severe, the dependent variables of (1) and (2) only include the position of the first virtual point.

2) Navigation: In the navigation environment, we redefine the state vector $\mathbf{s}[k]$ at time k as $(x_c, y_c, \delta x, \delta y)^T$, where $(x_c, y_c)^T$ is the centroid of the robot $(x_1, y_1)^T$, and $(\delta x, \delta y)^T$ is a vector pointing from the first point to the second point such that $(\delta x, \delta y)^T = (x_2 - x_1, y_2 - y_1)^T$. The decomposed incremental dynamics function $\mathbf{f}^n_{dec}$ can be expressed as

$$\Delta^n_{tr}[k] = \mathbf{f}^n_{tr}(\delta x[k], \delta y[k], \dot{x}_c, \dot{y}_c, v^n[k]) \tag{4}$$

$$\Delta^n_{rot}[k] = \mathbf{f}^n_{rot}(\delta x[k], \delta y[k], \omega^n[k]) \tag{5}$$

After a control input $(v^n, \omega^n)^T$ is applied, the total state increment $\Delta^n_{\mathbf{s}}[k]$ expressed as $(\Delta_x, \Delta_y, \Delta_{dx}, \Delta_{dy})^T$ will be the sum of the two components $\Delta^n_{tr}[k] + \Delta^n_{rot}[k]$.

The goal of learning $\mathbf{f}$ becomes that of learning $\mathbf{f}^n_{tr}$ and $\mathbf{f}^n_{rot}$, which form the decomposed incremental dynamics $\mathbf{f}^n_{dec}$. Note that $\Delta^n_{tr}[k]$ is in general dependent on x_c and y_c, too. However, we assume that the camera is viewing the scene approximately from above, so we can remove the dependency on the centroid location. Although $\mathbf{f}^n_{dec}$ is not exactly the same as the ground truth dynamics, $\mathbf{f}^n_{dec}$ can approximately capture the forced dynamics of the robot, and a well-designed closed-loop controller can compensate the mismatch during deployment.

In the real navigation environment, the Sphero BOLT will generally not reach the commanded linear velocity within one control step. Instead, an internal velocity controller will accelerate or decelerate the robot until it reaches the commanded velocity, which usually takes more than one control step. Since we do not have access to the design of this internal velocity controller, one way to approximate the translation dynamics is to add a velocity measurement as an independent variable. Therefore, Equations 4 include the velocity of the centroid $(\dot{x}_c, \dot{y}_c)^T$. Since the control command is applied every 10 frames of the camera, we estimate the centroid's velocity by retrieving another centroid position from the previous image frame acquired one step (1/60 s) before the control command, ensuring that the velocity estimate is up-to-date. Therefore, the centroid velocity can be estimated as the centroid difference multiplied by the frame rate of the camera, 60 fps.

C. Evaluation of Learned Dynamics Function

We collected data for dynamics learning in all three environments shown in Figure 1. Since the dynamics are decomposed into several parts, we separately collected datasets for learning each decomposed dynamics function. After the dynamics functions were obtained via supervised learning, we evaluated the learned dynamics in terms of their use for control. The first evaluation metric is the prediction error on a testing set. This is the standard evaluation metric for ML algorithms, where the whole dataset is divided into a training and testing sets, and the dynamics learned on the training set is evaluated on the testing set. The prediction error represents the one-step prediction error of the learned dynamics. We compared two different ML algorithms that are often used for learning dynamics functions: Locally Weighted Regression (LWR) and Gaussian Process Regression (GPR) [8]. In the following tables, RMSE is defined as root mean square error and MAE is defined as mean absolute error. There are two corresponding metrics, NRMSE and NMAE, that are defined as the respective metrics divided by the standard deviation in the testing dataset.

TABLE I

ONE-STEP PREDICTION ERROR ON THREE ENVIRONMENTS.

Models	Dyn. component	RMSE	NRMSE	Dyn. component	RMSE	NRMSE
			Object Handling Environment			
LWR	Rot	11.9644	1.1797	Trans	1.4184	1.0084
GPR		6.7326	**0.6638**		1.3523	**0.9613**
			Simulated Navigation Environment			
LWR	Rot	0.0714	0.0319	Trans	0.0484	**0.0052**
GPR		0.0035	**0.0016**		0.0849	0.0091
			Real Navigation Environment			
LWR	Rot	2.0535	1.0518	Trans	2.3017	0.4968
GPR		0.8669	**0.4440**		1.9516	**0.4212**

TABLE II

LONG-TERM PREDICTION ERROR OF THE GPR MODEL ON THREE ENVIRONMENTS.

Steps	Dyn. component	MAE (deg)	NMAE	Dyn. component	RMSE (pixel)	NRMSE
			Object Handling Environment			
10		0.8198	0.1086		2.0892	0.0468
20	Rot	1.5511	0.2591	Trans	2.7234	0.0696
30		2.7742	0.3824		3.3281	0.0896
			Simulated Navigation Environment			
10		0.0695	0.0034		2.4340	0.0349
20	Rot	0.1120	0.0055	Trans	6.8965	0.0989
30		0.3658	0.0179		11.9782	0.1717
			Real Navigation Environment			
10		8.7435	0.0743		35.6605	0.1478
20	Rot	15.5726	0.1980	Trans	61.3825	0.2418
30		25.2268	0.2319		70.7925	0.2820

Results shown in Table I indicate that the GPR model has the best one-step prediction performance on average for all three environments. However, precise one-step prediction is just one requirement for a good dynamics function. To better combine the learned dynamics with greedy and sequential control, the learned dynamics need to have reasonable prediction accuracy after multiple steps, up to the prediction horizon. This evaluation can be done by collecting an evaluation dataset that contains different trajectories of the robot with different horizons based on different values of control inputs. Then, we can use the learned dynamics $\hat{\mathbf{f}}^m_{dec}$ and $\hat{\mathbf{f}}^n_{dec}$ to roll out from the initial state of each trajectory and compute the prediction error at the prediction horizon.

Based on Table II, the prediction error grows when the prediction horizon increases. This error accumulation is often seen in learning-based control. Some previous works minimize the accumulated error by applying sequence-to-sequence models, but still the model accumulates error inevitably. Based on the comparison between ground-truth rollout and predicted rollout, we observed that the error also comes from the coupling effect of translation and rotation dynamics, as well as the sliding motion during the rolling of the Sphero BOLT. Therefore, in reality, we cannot obtain a perfect dynamics function with limited amount of data. Instead, we focused on developing a closed-loop controller that takes into account the uncertainties in the dynamics function while simultaneously steering the robot to the goal.

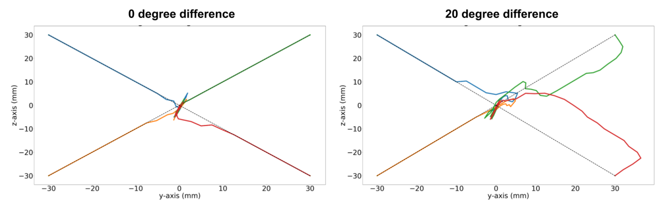

Fig. 3. Comparison between our image-based VS control (colored solid curve) and pose-based Cartesian space control (black dashed line). Cartesian-space control serves as an upper bound on performance for any VS algorithm, since the goal configuration of the robot in task space is given to it. Both plots show the results of four different starting poses (4 corners of a square). The starting poses in the left plot do not have rotational deviation with respect to the goal pose, whereas the ones in the right plot have rotational deviation of 20 degrees. Although the two controllers choose different routes, they ultimately reach the goal (square center).

D. Evaluation of VS Control with Compact State on the Holonomic System

In this subsection, we report the performance of VS control in the object handling environment using the proposed compact state. We combine the learned dynamics described in the previous section with the greedy controller to move the box to a designated goal pose. We obtain the interaction matrix at the goal pose and use this interaction matrix to determine the moving direction at each control step.

The comparison baseline was a pose-based Cartesian control without any consideration of perception. Since we can record the ground-truth goal configuration of the robot when we define the goal pose of the box, using its precise joint encoders, the pose-based Cartesian control is simply a proportional control in Cartesian space. This control algorithm serves as an upper bound on performance for any VS algorithm, since the true goal 3D pose is given to it directly and the industrial robot has excellent repeatability (0.1 mm).

Figure 3 shows the trajectories of both controllers from different starting poses (four corners of a square) with the same goal pose (the center of the square). The left plot shows the case where the starting poses do not have rotational deviations with respect to the goal pose, whereas the right plot shows trials with a 20-degree rotational deviation. The trajectories of the pose-based Cartesian controller are the dashed black lines and the ones from our algorithm are the solid colored curves. Although the two controllers choose different routes, the results show that our algorithm can still reach the goal from different starting poses. Since we only estimate the interaction matrix at the goal point in these test cases, the trajectory might not be optimal. However, in this experiment, our goal was to show that a greedy VS controller using the learned dynamics can accomplish goal-reaching tasks on holonomic systems without knowing the tracked object's CAD model and without any camera calibration.

Table III shows evaluation results of our algorithm on 50 different starting poses with y- and z-axis deviations ranging from -30 mm to 30 mm and rotation angle deviations ranging from -30 degrees to 30 degrees. Since we can accurately obtain the ground truth about the robot's configuration from the encoders, the distance between the final pose of the trajectory and the goal pose can be measured in Cartesian space. Because the robot has good repeatability, the error of the pose-based Cartesian space controller is close to zero. Therefore, the error in the table shows how close the VS controller is to this lower bound. These results demonstrate that the proposed algorithm can accurately reach the target pose using only image information.

TABLE III
PERFORMANCE OF THE GREEDY CONTROLLER WITHIN A HORIZON OF 50 CONTROL STEPS IN THE OBJECT HANDLING ENVIRONMENT

Task	Dyn. component	RMSE (mm)	NRMSE	Dyn. component	MAE (deg)	NMAE
Pure Trans	Trans	0.3116	0.0119	Rot	N/A	
Trans + Rot		1.5521	0.0592		0.2691	0.0110

E. Evaluation of iLQR with Compact State on the Non-Holonomic System

We experimentally validated that the greedy controller described in Section VI-D cannot control the mobile robot to do certain goal-reaching tasks because of its nonholonomic constraints. In contrast to the greedy controller, a sequential controller such as iLQR ([9]) was experimentally proven to achieve most of the goal-reaching tasks in the simulated navigation environment. Therefore, in this subsection, we report the evaluation of the effectiveness of using compact state and learned dynamics in designing an iLQR-based VS controller in the real-world navigation environment.

We consider two kinds of goal-reaching tasks: turn-in-place tasks and general tasks. For turn-in-place tasks, the robot is asked to turn its heading to align with the given target heading. Task A is defined as 90 degree turn-in-place and task B is defined as 180 degree turn-in-place.

Regarding general goal-reaching tasks, we set the goal state as the center of the table with the heading pointing at the positive x axis and placed the robot at a random position on the table with a random heading. Since the robot needs to turn and move at the same time in order to reach the goal, this general task can evaluate the ability of the controller to jointly determine two control inputs v^n and ω^n, under the nonholonomic constraint.

Across different goal-reaching tasks, we select two difficulty levels of the scenarios. The simpler among them is to place the robot at the negative x axis (on the left-hand side of the goal state) pointing at -45 degrees, which is denoted as task C. The more difficult task is to place the robot at the negative y axis (the lower side of the goal state) with the same heading as the goal state, which is denoted as task D. We differentiate the level by the magnitude of the projection of the initial feedback error $\mathbf{e}[0] = \mathbf{s}[0] - \mathbf{s}^*$ on the initial heading vector $(\delta x, \delta y)^T$. The simpler task has a non-zero projection and the more difficult one has a nearly zero projection. The more difficult one is commonly known as the parallel parking scenario, where a nonholonmic robot cannot reach the goal state by simply moving sideways.

We believe that PBVS can serve as a comparable algorithm for our proposed method. However, PBVS needs to estimate the robot's 3D pose beforehand and different pose estimators

TABLE IV

COMPARISON BETWEEN PBVS AND OUR METHOD WITHIN A HORIZON OF 50 CONTROL STEPS IN THE SIMULATOR

Task	Method	Dyn. component	MAE (deg)	NMAE	Dyn. component	RMSE (pixel)	NRMSE
A	PBVS	Rot	0.2153	0.0010	Trans	N/A	
	Ours		1.0779	0.0051			
B	PBVS	Rot	0.3807	0.0018	Trans	N/A	
	Ours		2.0745	0.0098			
C	PBVS	Rot	0.7874	0.0254	Trans	0.2119	0.0068
	Ours		1.1373	0.0358		0.8134	0.0256
D	PBVS	Rot	0.8224	0.0064	Trans	0.2076	0.0016
	Ours		1.0143	0.0079		0.7971	0.0062

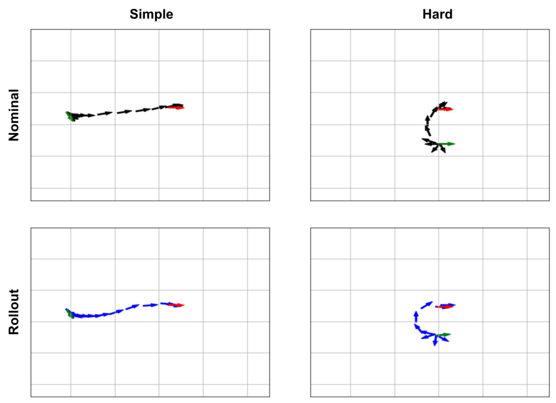

Fig. 4. Comparison between nominal trajectory from iLQR and closed-loop trajectory in the real-world. The iLQR controller will bring the misaligned state back to the nominal trajectory and eventually reach the goal.

may have different prediction errors. To have a fair comparison, we compare our method with PBVS in the simulator since the robot's ground-truth 3D pose and dynamics can be directly obtained. Therefore, PBVS is regarded as the performance upper bound of our algorithm.

Table IV shows the error between the final pose of the robot and the goal pose after the iLQR controller is applied over 50 steps. Based on Table IV, our algorithm has comparable results to PBVS. Since the best we can do is obtain ground-truth pose and dynamics, this table shows that our developed algorithm is close to this upper bound.

To evaluate overall performance on nonholonomic systems in the real-world, we combined the compact state descriptor, the learned dynamics, and the controller. Since closed-loop controllers are known to regulate dynamics well and are generally robust to uncertainties in the dynamics, the focus of this section is to examine whether a closed-loop controller can still achieve the goal given imperfect dynamics.

Based on Figure 4, the rollout (actual) and nominal (planned) trajectories are somewhat different from each other, but the goal can still be reached via the feedback control law. The reason is that when the state deviates from the nominal trajectory due to uncertainties in the dynamics, the controller will act to bring the system's state back to the nominal trajectory through the optimal control gains. Based on Table V, the closed-loop iLQR controller can not only bring the robot to the goal, but also minimizes the accumulated error in the learned dynamics function. After 30 horizon steps, the error is largely minimized compared to the prediction error after 30 steps shown in Table II.

TABLE V

PERFORMANCE OF THE iLQR CONTROLLER WITHIN A HORIZON OF 30 CONTROL STEPS IN THE REAL-WORLD

Task	Dyn. component	MAE (deg)	NMAE	Dyn. component	RMSE (pixel)	NRMSE
A	Rot	4.6632	1.4306	Trans	N/A	
B		7.4607	3.7076			
C		11.8958	1.1052		23.9420	0.1742
D		15.4372	0.0945		19.5438	0.1276

VII. CONCLUSION AND FUTURE WORK

The proposed method for VS control of robots using images from an uncalibrated camera constructs compact state representations of the robot's configuration and uses transition dynamics learned from collected execution traces to compute control velocities to reach a desired goal state identified directly by its image. The method was verified experimentally for planar motion of both a fully actuated manipulator arm as well as an underactuated mobile robot with nonholonomic constraints. The key step of the proposed method is the estimation of a homography transform between the image positions of distinct keypoints belonging to the robot in the current image and those in a reference image, which can be done quickly and robustly even when not the same set of keypoints is observed at each time step, making it robust to noise and variations in illumination. This approach is based on the assumption that the tracked keypoints are coplanar, which is largely justified for overhead placement of the camera. In future work, we plan to relax this assumption, possibly using more than one camera, and also extending the method to full 3D spatial motion.

REFERENCES

[1] F. Chaumette, S. Hutchinson, and P. Corke, "Visual servoing," *Handbook of Robotics, 2nd ed.*, pp. 841–866, 2016.

[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, "Mobile robots," *Robotics: Modelling, Planning and Control*, pp. 469–521, 2009.

[3] K. Usher, P. Ridley, and P. Corke, "Visual servoing of a car-like vehicle-an application of omnidirectional vision," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 3. IEEE, 2003, pp. 4288–4293.

[4] X. Zhang, Y. Fang, B. Li, and J. Wang, "Visual servoing of nonholonomic mobile robots with uncalibrated camera-to-robot parameters," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 1, pp. 390–400, 2016.

[5] C. Wang, Y. Mei, Z. Liang, and Q. Jia, "Dynamic feedback tracking control of non-holonomic mobile robots with unknown camera parameters," *Transactions of the Institute of Measurement and Control*, vol. 32, no. 2, pp. 155–169, 2010.

[6] H. Chen, S. Ding, X. Chen, L. Wang, C. Zhu, and W. Chen, "Global finite-time stabilization for nonholonomic mobile robots based on visual servoing," *International Journal of Advanced Robotic Systems*, vol. 11, no. 11, p. 180, 2014.

[7] S. Benhimane and E. Malis, "Homography-based 2d visual tracking and servoing," *The International Journal of Robotics Research*, vol. 26, no. 7, pp. 661–676, 2007.

[8] A. Chiuso and G. Pillonetto, "System identification: A machine learning perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 281–304, 2019.

[9] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *First International Conference on Informatics in Control, Automation and Robotics*, vol. 2. SciTePress, 2004, pp. 222–229.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.