

Control with Patterns: A D-learning Method

Anonymous Author(s)

Affiliation

Address

email

Abstract: Learning-based control policies are widely used in various tasks in the field of robotics and control. However, formal (Lyapunov) stability guarantees for learning-based controllers with nonlinear dynamical systems are challenging to obtain. We propose a novel control approach, namely Control with Patterns (CWP), to address the stability issue over data sets corresponding to nonlinear dynamical systems. For data sets of this kind, we introduce a new definition, namely exponential attraction on data sets, to describe nonlinear dynamical systems under consideration. The problem of exponential attraction on data sets is converted to a pattern classification one based on the data sets and parameterized Lyapunov functions. Furthermore, D-learning is proposed as a method for performing CWP without knowledge of the system dynamics. Finally, the effectiveness of CWP based on D-learning is demonstrated through simulations and real flight experiments. In these experiments, the position of the multicopter is stabilized using only real-time images as feedback, which can be considered as an Image-Based Visual Servoing (IBVS) problem.

Keywords: Lyapunov Methods, Reinforcement Learning, Control with Patterns, D-learning, Visual Servoing

1 Introduction

In a data-rich age, a system is often under operation when measurements of system inputs and outputs are accessible for collection through inexpensive and numerous information-sensing devices. Based on the input and output data, a direct way is often to model the dynamical system according to the first principles. Then, existing methods are used to analyze the stability or design controllers for the identified system.

However, there exist two difficulties. First, it is not easy to get the true form of the considered system, so the approximation may not be satisfied. Secondly, except for only a few experts, the approximated model may still be hard to handle with existing model-based methods.

The development of deep learning and reinforcement learning [1],[2] has led to new advances in these difficulties [3],[4],[5]. The advancement of deep learning and reinforcement learning has significantly contributed to the development of neural network controllers for robotic systems [6],[7],[8]. For further discussion of related works, please refer to Appendix A.

Despite the impressive performance of these controllers, many of these works lack critical stability guarantees that are essential for safety-critical applications. To overcome this lack, Lyapunov stability [20] in control theory provides a well-known framework for ensuring closed-loop stability of nonlinear dynamical systems. The core concept of this theory is the Lyapunov function, a scalar function whose value decreases along the closed-loop trajectory of the system. This function

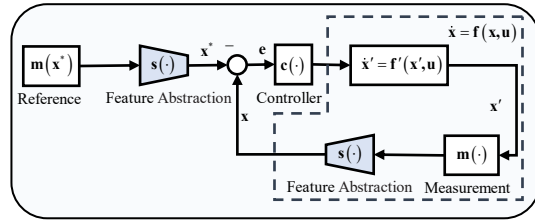


Figure 1: Closed-loop system by CWP.

40 demonstrates the process by which the system transitions from the system from any state within the
 41 Region of Attraction (ROA) to a stable equilibrium.

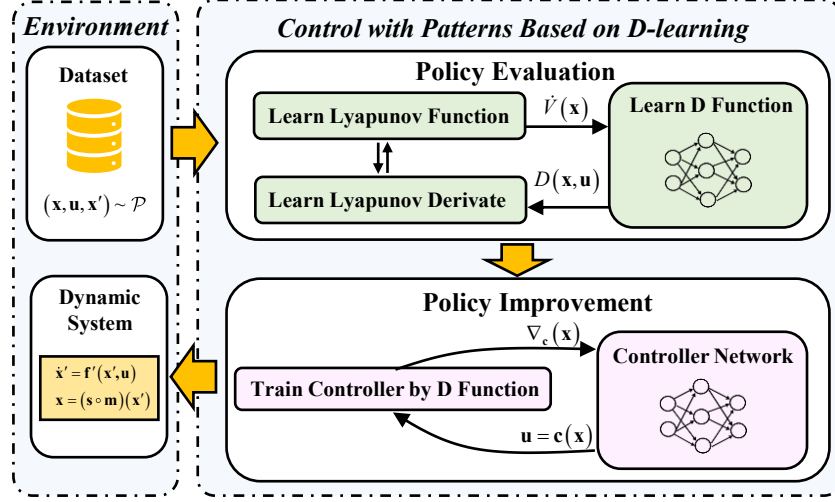


Figure 2: Overview of our method. The main contents consist of two parts. For the policy evaluation step, the Lyapunov function and D function is updated by solving (13). After learning the D function, we train the CWP controller by solving (14).

42 Previous studies [9],[10] that integrate deep learning and Lyapunov control methods have primarily
 43 furnished guarantees for state feedback control based on structured information (e.g., the state of
 44 a linear time-invariant system). Our work addresses the more challenging but practically relevant
 45 concern of unstructured information-based feedback control by identifying and overcoming the lim-
 46 itations of existing approaches to synthesize, and certify controllers for real-world applications. In
 47 order to demonstrate the effectiveness of our method on real robotic systems, we design a model-free
 48 flight controller that can (1) stabilize a hovering multicopter with only images as feedback, similar to
 49 visual servo controllers; (2) outperform reinforcement learning; and (3) provide Lyapunov stability
 50 guarantees.

51 Our key contributions are:

- 52 • We propose a Control with Patterns (CWP) approach for the stability issue of dynamical sys-
 53 tems, which transforms the controller design problem into a pattern classification problem. CWP
 54 represents a novel framework related to Lyapunov function learning, that can be used to develop
 55 model-free controllers for general dynamical systems.
- 56 • We propose D-learning, which parallels to Q-learning [11] in Reinforcement Learning (RL) to ob-
 57 tain both Lyapunov function and its derivative. Unlike existing Lyapunov function learning methods
 58 relying on controlled models or their approximation with neural networks [12],[13], the system dy-
 59 namics are encoded into the so-called D function depending on actions. This allows one to perform
 60 CWP without any knowledge of the system dynamics.
- 61 • The results of the simulation platform and real flight experiments demonstrate that our approach
 62 can stabilize a multicopter with only real-time images as feedback. Furthermore, the controller
 63 trained by D-learning exhibits superior performance to the controller trained by RL.

64 2 Problem Formulation

65 Consider the following autonomous system

$$\begin{aligned}\dot{\mathbf{x}}' &= \mathbf{f}'(\mathbf{x}', \mathbf{u}) = \mathbf{f}'(\mathbf{x}', \mathbf{c}(\mathbf{x})) \\ \mathbf{x} &= \mathbf{s}(\mathbf{m}(\mathbf{x}'))\end{aligned}$$

where $\mathbf{x}' \in \mathcal{D}' \subseteq \mathbb{R}^{n'}$ is original state unavailable to measure, $\mathbf{m}(\mathbf{x}') \in \mathcal{M}$ is a measurement in the form of unstructured data such as images, $\mathbf{s}(\cdot)$ is a designed *feature* selection function to code the measurement to a vector $\mathbf{x} \in \mathcal{D} \subseteq \mathbb{R}^n$, and $\mathbf{u} = \mathbf{c}(\mathbf{x})$ is the control policy. We aim to focus on \mathbf{x} rather than \mathbf{x}' , namely considering the following autonomous system

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{s}}{\partial \mathbf{m}} \frac{\partial \mathbf{m}}{\partial \mathbf{x}'} \mathbf{f}'(\mathbf{x}', \mathbf{u}) \triangleq \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (1)$$

where $\mathbf{f} : \mathcal{D} \rightarrow \mathbb{R}^n$, $\mathbf{x}(0) = \mathbf{x}_0 \in \mathcal{D}$. From observing the controlled system (1), we can obtain the data set

$$\mathcal{P}_u = \{(\dot{\mathbf{x}}_i, \mathbf{x}_i), i = 1, \dots, N\}. \quad (2)$$

We prepare to solve the *exponential attraction* (See Definition B.2) problem using the Lyapunov method. The Lyapunov function for the data set (2) is supposed to have the following form

$$V(\mathbf{x}) = \mathbf{g}(\mathbf{x})^T \boldsymbol{\theta}_g \quad (3)$$

where $V(\mathbf{0}_{n \times 1}) = 0$, $V : \mathcal{D} / \{\mathbf{0}_{n \times 1}\} \rightarrow \mathbb{R}_+$ and $\boldsymbol{\theta}_g \in \mathcal{S}_g \subseteq \mathbb{R}^{l_1}$. The set \mathcal{S}_g is used to guarantee that the function $V(\mathbf{x})$ is a Lyapunov function. The derivative of $V(\mathbf{x})$ yields

$$\dot{V}(\mathbf{x}) = \left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \dot{\mathbf{x}} \right)^T \boldsymbol{\theta}_g. \quad (4)$$

We hope that the derivative satisfies

$$\dot{V}(\mathbf{x}) \leq -W(\boldsymbol{\theta}_h, \mathbf{x}) \quad (5)$$

where $W(\boldsymbol{\theta}_h, \mathbf{x})$ is also a Lyapunov function, which can be further written as

$$W(\boldsymbol{\theta}_h, \mathbf{x}) = \mathbf{h}(\mathbf{x})^T \boldsymbol{\theta}_h$$

where $\boldsymbol{\theta}_h \in \mathcal{S}_h \subseteq \mathbb{R}^{l_2}$. Similarly, the set \mathcal{S}_h is used to guarantee that the function $W(\mathbf{x})$ is a Lyapunov function as well.

For the data set (2), suppose that we have

$$\left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i \right)^T \boldsymbol{\theta}_g \leq -\mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\theta}_h \quad (6)$$

where $i = 1, \dots, N$. Then, in the following, based on (6), we show that the equilibrium state $\mathbf{x} = \mathbf{0}_{n \times 1}$ is *exponentially attractive* on the data set \mathcal{P} in Theorem 2.1. For proof, please see Appendix C.2.

Theorem 2.1. *Under Assumptions C.1-C.5, for the autonomous system (1), if there exist parameter vectors $\boldsymbol{\theta}_g \in \mathcal{S}_g$ and $\boldsymbol{\theta}_h \in \mathcal{S}_h$ such that (6) holds for the data set \mathcal{P} , then the equilibrium state $\mathbf{x} = \mathbf{0}_{n \times 1}$ is exponentially attractive on the data set \mathcal{P} .*

Consequently, according to Theorem 2.1, the exponential attraction problem is converted to make the inequality (6) hold. The inequality (6) is rewritten as

$$\mathbf{y}_i^T \boldsymbol{\theta} \geq 0, i = 1, \dots, N \quad (7)$$

where

$$\mathbf{y}_i = - \left[\left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i \right)^T \quad \mathbf{h}(\mathbf{x}_i)^T \right]^T$$

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_g \\ \boldsymbol{\theta}_h \end{bmatrix} \in \mathcal{S} \triangleq \mathcal{S}_g \times \mathcal{S}_h.$$

Formally, according to Theorem 2.1, we construct the *Control with Patterns* (CWP) problem formulation represented as follows

$$\text{Design } \mathbf{u} \in \mathcal{U} \text{ and find } \boldsymbol{\theta} \in \mathcal{S} \text{ to make (7) hold on the data set (2)} \quad (8)$$

This problem is a *pattern classification* [14] problem. Here, \mathbf{y}_i is the compound *features* which describe the stability *pattern* for the autonomous system (1). As a result, $f(\boldsymbol{\theta}) = \mathbf{y}^T \boldsymbol{\theta}$ can be considered as a *linear discriminant function* [14]. So far, we turned the problem of system stabilization (6) into a problem of pattern classification (8).

96 3 Control with Patterns based on D-learning

97 After formulating the CWP problem (8), we are going to consider how to construct the model-free
98 controller based on data sets. To this end, we will design CWP controller based on a proposed
99 D-learning method.

100 3.1 Control with Patterns

101 In order to solve the CWP problem (8), we solve the following optimization

$$\begin{aligned} & \min_{\eta, \theta_g \in \mathcal{S}_g, a > 0, \mathbf{c}} \quad wa - \eta \\ & \text{s.t.} \quad - \left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i(\mathbf{c}) \right)^T \boldsymbol{\theta}_g - W(\boldsymbol{\theta}_g, \mathbf{x}_i) \geq 0 \\ & \quad F_g(\boldsymbol{\theta}_g) \leq a \end{aligned} \quad (9)$$

102 where $i = 1, \dots, N$, $F_g(\cdot)$ is a constraint on $\boldsymbol{\theta}_g$, $W(\boldsymbol{\theta}_g, \mathbf{x})$ is a Lyapunov function mentioned in
103 (5). In the following for simplicity, let $W(\boldsymbol{\theta}_g, \mathbf{x}) = \eta \|\mathbf{x}\|^2$. An iterative procedure for solving the
104 inequality (9) may be used, including *policy evaluation* and *policy improvement*.

105 • **Initialization.** Select any admissible (i.e., stabilizing) control \mathbf{c}_0 , $k = 0$.

106 • **Policy Evaluation Step.** Under \mathbf{c}_k , at state \mathbf{x}_i , the control is $\mathbf{u} = \mathbf{c}_k(\mathbf{x}_i) \in \mathcal{U}$, resulting in
107 $\dot{\mathbf{x}}_i(\mathbf{c}_k) \in \mathcal{D}$. Determine the solution $\boldsymbol{\theta}_{g,k}$, $a > 0$, η_k by

$$\begin{aligned} & \min_{\eta, \theta_g \in \mathcal{S}_g, a > 0} \quad wa - \eta \\ & \text{s.t.} \quad - \left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i(\mathbf{c}_k) \right)^T \boldsymbol{\theta}_g - \eta \|\mathbf{x}_i\|^2 \geq 0 \\ & \quad F_g(\boldsymbol{\theta}_g) \leq a \end{aligned} \quad (10)$$

108 where $i = 1, \dots, N_k$.

109 • **Policy Improvement Step.** Determine an improved policy using

$$\begin{aligned} & \min_{\mathbf{c}, \eta} \quad -\eta \\ & \text{s.t.} \quad - \left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i(\mathbf{c}) \right)^T \boldsymbol{\theta}_{g,k} - \eta \|\mathbf{x}_i\|^2 \geq 0 \end{aligned} \quad (11)$$

110 where $i = 1, \dots, N_k$.

111 By fixing \mathbf{x}_i for every step k , the iteration can be chosen to stop after sufficient steps if $wa_k - \eta_k$
112 and $-\eta_k$ are nearly not changed. This is because the iterative procedure is in fact used to solve the
113 optimization (9) with the coordinate descent [15].

114 3.2 Control with Patterns Based on D-Learning

115 Unfortunately, in the *Policy Improvement Step* (10), one requires knowledge of the system dynamics
116 $\dot{\mathbf{x}}_i(\mathbf{c})$. To avoid knowing any of the system dynamics, similar to Q-Learning [29] in the field of RL,
117 we can rewrite $\dot{V}(\mathbf{x})$ in (4) as

$$D(\mathbf{x}, \mathbf{u}) = \left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u}) \right)^T \boldsymbol{\theta}_g$$

118 where (1) is utilized. We call it the D function as it is the *derivative* of the Lyapunov function
119 and it is expected to be *decreased*. If one obtains $D(\mathbf{x}, \mathbf{u})$ by learning directly, then the use of the
120 input coupling function is avoided. In the nonlinear case, one assumes that the value of $D(\mathbf{x}, \mathbf{u})$ is
121 sufficiently smooth. Then, according to the Weierstrass higher order approximation theorem, there
122 exists a dense basis set $\{\varphi_i(\mathbf{x}, \mathbf{u})\}$ such that

$$D(\mathbf{x}, \mathbf{u}) = \sum_{i=1}^{\infty} \theta_i \varphi_i(\mathbf{x}, \mathbf{u}) = \sum_{i=1}^L \theta_i \varphi_i(\mathbf{x}, \mathbf{u}) + \sum_{i=L+1}^{\infty} \theta_i \varphi_i(\mathbf{x}, \mathbf{u}) \triangleq \boldsymbol{\theta}_d^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{u}) + \varepsilon_L(\mathbf{x}, \mathbf{u})$$

where basis vector $\boldsymbol{\theta}_d = [\theta_1 \ \theta_2 \ \cdots \ \theta_L]^T$, $\boldsymbol{\phi}(\mathbf{x}, \mathbf{u}) = [\varphi_1(\mathbf{x}, \mathbf{u}) \ \varphi_2(\mathbf{x}, \mathbf{u}) \ \cdots \ \varphi_L(\mathbf{x}, \mathbf{u})]^T$ and ε_L converges uniformly to zero as the number of terms retained $L \rightarrow \infty$.

It is expected to make $D(\mathbf{x}, \mathbf{u}) - \dot{V}(\mathbf{x})$ as small as possible. Mathematically, one has to decrease b as small as possible with the following constraint

$$\left| \boldsymbol{\theta}_d^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{c}_k(\mathbf{x}_i)) - \boldsymbol{\theta}_g^T \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i(\mathbf{c}_k(\mathbf{x}_i)) \right| \leq b$$

where $i = 1, \dots, N_k$, $k = 1, \dots, M$.

On the other hand, (5) is rewritten as

$$\boldsymbol{\theta}_d^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{u}) \leq -\eta \|\mathbf{x}\|^2.$$

Furthermore, the inequality (7) is rewritten as

$$\mathbf{y}_i^T \boldsymbol{\theta}' \geq 0, i = 1, \dots, N \quad (12)$$

where

$$\begin{aligned} \mathbf{y}_i' &= - \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{u}_i)^T & \mathbf{h}(\mathbf{x}_i)^T \end{bmatrix}^T \\ \boldsymbol{\theta}' &= \begin{bmatrix} \boldsymbol{\theta}_d \\ \boldsymbol{\theta}_h \end{bmatrix} \in \mathcal{S}' \triangleq \mathbb{R}^L \times \mathcal{S}_h. \end{aligned}$$

With the D function, iterative procedures for solving the inequality (9) should be rewritten, including *policy evaluation* and *policy improvement*.

• **Initialization.** Select any admissible (i.e., stabilizing) control \mathbf{c}_0 , $k = 0$.

• **Policy Evaluation Step (Based on D-learning).** Under \mathbf{c}_k , at state \mathbf{x}_i , the control is $\mathbf{u} = \mathbf{c}_k(\mathbf{x}_i) \in \mathcal{U}$, resulting in $\dot{\mathbf{x}}_i(\mathbf{c}_k) \in \mathcal{D}$. Determine the solution $\boldsymbol{\theta}_{g,k}, \boldsymbol{\theta}_{d,k}, a_k, b_k > 0, \eta_k \in \mathbb{R}$ by

$$\begin{aligned} \min_{\boldsymbol{\theta}_d \in \mathbb{R}^L, \boldsymbol{\theta}_g \in \mathcal{S}_g, a, b > 0, \eta \in \mathbb{R}} \quad & -\eta + w_1 a + w_2 b \\ \text{s.t.} \quad & -\boldsymbol{\theta}_d^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{c}_k(\mathbf{x}_i)) - \eta \|\mathbf{x}_i\|^2 \geq 0 \\ & \left| \boldsymbol{\theta}_d^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{c}_j(\mathbf{x}_i)) - \boldsymbol{\theta}_g^T \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i(\mathbf{c}_j(\mathbf{x}_i)) \right| \leq b \\ & F_g(\boldsymbol{\theta}_g) \leq a \end{aligned} \quad (13)$$

where $w_1, w_2 > 0$ are weights, $\dot{\mathbf{x}}_i(\mathbf{c}_j(\mathbf{x}_i)) = \mathbf{f}(\mathbf{x}_i, \mathbf{c}_j(\mathbf{x}_i))$, $j = 0, \dots, k$, $i = 1, \dots, N_k$.

• **Policy Improvement Step (Based on D-learning).** Determine an improved policy using

$$\begin{aligned} \min_{\mathbf{c}, \eta \in \mathbb{R}} \quad & -\eta \\ \text{s.t.} \quad & -\boldsymbol{\theta}_{d,k}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{c}(\mathbf{x}_i)) - \eta \|\mathbf{x}_i\|^2 \geq 0 \end{aligned} \quad (14)$$

where $i = 1, \dots, N_k$.

4 Simulations and Experiments

In this section, simulations and experiments demonstrate that the CWP-based controller can stabilize a hovering multicopter with only images as feedback, which can be considered as an IBVS [16] problem.

4.1 Simulations Design

4.1.1 Problem Formulation of Visual Servoing

Since the camera is fixed to the body of the multicopter, based on Semi-Autonomous Autopilots (SAAs), the plant [17] can be modeled as

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \mathbf{I} &= \mathbf{m}_c(\mathbf{r}) \end{aligned} \quad (15)$$

where $\mathbf{r}, \mathbf{v} \in \mathbb{R}^3$ indicate the pose and velocity of the multicopter, and $\mathbf{I} = \mathbf{m}_c(\mathbf{r})$ represents the mapping from the position \mathbf{r} of the multicopter to the image $\mathbf{I} \in \mathcal{I}$ taken by the camera.

For IBVS, visual servoing is a minimization problem between the image features $\mathbf{s}(\mathbf{r})$ extracted from the current pose \mathbf{r} and the desired features \mathbf{s}^* from the desired pose \mathbf{r}^* . Then, the controller is decided by

$$\begin{aligned}\mathbf{e} &= \mathbf{s}(\mathbf{I}) - \mathbf{s}(\mathbf{I}^*) \\ \mathbf{v} &= \mathbf{c}(\mathbf{e})\end{aligned}\tag{16}$$

where $\mathbf{s}(\cdot) \in \mathcal{S}$ is a designed feature selection function, such as neural networks, to code the measurement to a vector in a latent space \mathcal{S} ; $\mathbf{v} = \mathbf{c}(\mathbf{e})$ represents the controller based on CWP, and its input \mathbf{e} is the error between the features of current image $\mathbf{I} \in \mathcal{I}$ and the desired image \mathbf{I}^* .

To extract the features from the current image \mathbf{I} , we use deep metric learning methods, suggested by the work [18], to train the feature selection function. More details about feature extraction can be obtained in Appendix D.

4.1.2 D-learning Controller Design

We train the D-learning controller based on the latent Shape \mathcal{S} . The Lyapunov function is designed as a quadratic function as

$$V(\mathbf{e}) = \mathbf{e}^T \mathbf{P} \mathbf{e} = \mathbf{g}^T(\mathbf{e}) \boldsymbol{\theta}_g\tag{17}$$

where $\mathbf{e} = \mathbf{s}(\mathbf{I}) - \mathbf{s}(\mathbf{I}^*)$, $\mathbf{g}(\mathbf{e}) = [\mathbf{e}^T \otimes \mathbf{e}]^T$ (\otimes represents Kronecker product), and $\boldsymbol{\theta}_g = \text{vec}(\mathbf{P})$, which represents the vectorization of the matrix \mathbf{P} .

Using the Lyapunov function (17), the CWP controller $\mathbf{u} = \mathbf{c}(\mathbf{e})$ is given in Algorithm 1, in which the D function $D(\mathbf{e}, \mathbf{u})$ and the CWP controller $\mathbf{u} = \mathbf{c}(\mathbf{e})$ is both designed as a 4-layer perceptron, with ReLU activations after each hidden layer.

Algorithm 1: Control with Patterns Based on D-learning with Constraints

Input: The data set generated by any admissible (i.e., stabilizing) control $\mathbf{u} = \mathbf{c}_0(\mathbf{e})$

Output: CWP controller $\mathbf{u} = \mathbf{c}_k(\mathbf{e})$

1 Initialization. Select any admissible (i.e., stabilizing) control policy parameters $\mathbf{u} = \mathbf{c}_0(\mathbf{e})$, D-function parameters $\boldsymbol{\theta}_{d,0}$, Lyapunov function parameters $\boldsymbol{\theta}_{g,0}$, and data set $\mathcal{P} = \{(\mathbf{e}_i, \mathbf{u}_i, t_i), i = 1, \dots, N\}$;

2 **while** stopping criterion not satisfied $\eta > 0$ **do**

3 Calculate the parameter $\boldsymbol{\theta}_g$ of the Lyapunov function by solving optimization problem

$$\begin{aligned}166 \quad & \min_{\boldsymbol{\theta}_g \in \mathcal{S}_g, a > 0} \quad wa - \eta \\ & \text{s.t.} \quad \begin{aligned} & -(\mathbf{g}(\mathbf{e}_{i+1})^T \boldsymbol{\theta}_g - \mathbf{g}(\mathbf{e}_i)^T \boldsymbol{\theta}_g) - \eta(t_{i+1} - t_i) \|\mathbf{e}_i\|^2 \geq 0 \\ & F_g(\boldsymbol{\theta}_g) \leq a \end{aligned} \end{aligned}$$

where $i = 1, \dots, N - 1$, and $F_g(\cdot)$ represents the constraints on the variable $\boldsymbol{\theta}_g$;

4 Estimate the Lyapunov derivative function $\dot{V}(\mathbf{e}_i) = V(\mathbf{e}_{i+1}) - V(\mathbf{e}_i)/(t_{i+1} - t_i)$;

5 Update D-function by (13), where $w_1, w_2 > 0$ are weights, $i = 1, \dots, N - 1$;

6 Determine an improved policy $\mathbf{u} = \mathbf{c}_k(\mathbf{e})$ by solving optimization problem (14);

7 **end**

4.2 Simulations Results

To verify the effectiveness of our control algorithm in the IBVS task, we first perform experimental validation in a simulation environment constructed using RflySim¹.

We sample the camera position in a dimension of 22 m \times 8 m centered on the desired position. Then, the multicopter departs from a set start position and arrives at the desired position by the Position-Based Visual Servoing (PBVS) [16] method based on the position information. By sampling with

¹<https://rflysim.com/>

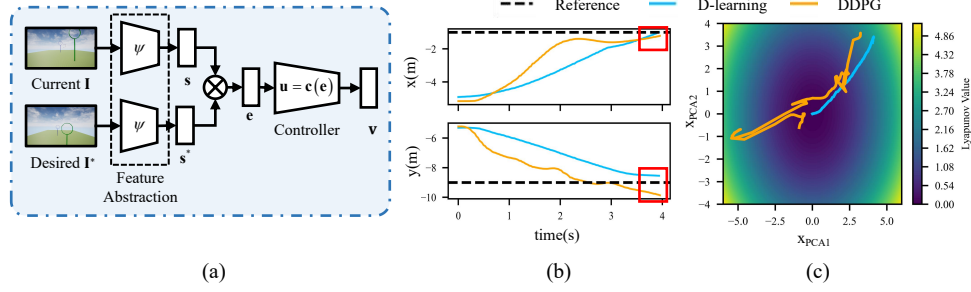


Figure 3: The simulation experiments and results. (a) The control system of multicopter. Given the desired image \mathbf{I}^* and the current image \mathbf{I} acquired from the camera, the feature error $\mathbf{s}(\mathbf{I}) \in \mathcal{S} \subseteq \mathbb{R}^{32}$ is solved by neural network $\psi : \mathcal{I} \rightarrow \mathbb{R}^{32}$. The encoding error $\mathbf{e} = \mathbf{s}(\mathbf{I}) - \mathbf{s}(\mathbf{I}^*)$ is used as an input to the controller $\mathbf{u} = \mathbf{c}(\mathbf{e})$. The output of the controller is velocity $\mathbf{v} \in \mathbb{R}^3$. (b) Performance evaluations of the servo error $\Delta \mathbf{r} = \mathbf{r} - \mathbf{r}^*$ on the simulation environment compared between the controller trained by D-learning and DDPG. Red square shows that D-learning controller have smaller error than DDPG controller. (c) PCA projection of the Lyapunov function (17) learned for the system (15), overlaid with trajectories of the system controlled by the D-learning controller and DDPG controller, which shows that D-learning controller have better stability guarantees than DDPG controller.

173 equal spacing, 301 trajectories are captured. Based on the collected data tuple $(\mathbf{r}, \mathbf{u}, \mathbf{I}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times$
 174 \mathcal{I} , we first train the network $\psi : \mathcal{I} \rightarrow \mathcal{S}$ to obtain $\mathbf{s}(\mathbf{I}) \in \mathbb{R}^{32}$. Then, we use Algorithm 1 to train
 175 the controller based on D-learning. Finally, we replace the PBVS controller with the D-learning
 176 controller, the experimental results are shown in Fig.3(a). The CWP controller can stabilize the
 177 multicopter using only images as feedback.

178 As a comparison, we also train the RL controller. We consider the collected trajectory data as a
 179 replay buffer, and use the Deep Deterministic Policy Gradient (DDPG) [19] algorithm to train an
 180 actor as a controller. The comparison between the D-learning controller and the DDPG controller
 181 is shown in Fig.3(b), in which the DDPG controller, although it can also converge to the reference
 182 point, the error is larger than that of the D-learning controller and the Lyapunov function fails to
 183 achieve sustained convergence. This result manifests that the D-learning controller provides more
 184 reliable stability guarantees than the RL controller.

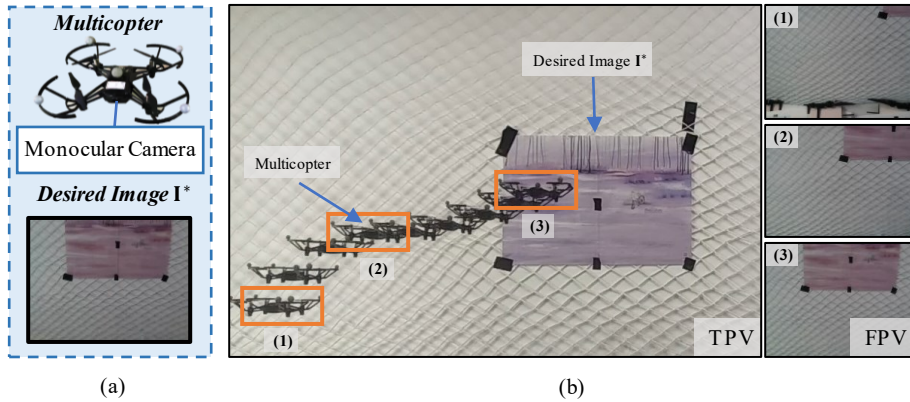


Figure 4: The real flight experiments and results. (a) On the real flight experiments, we use DJI Tello EDU, which features a front-facing camera. The desired image \mathbf{I}^* of IBVS is centered on a drawing. (b) On the left is the Third-Person View (TPV) and on the right is the onboard camera's First-Person View (FPV). The multicopter positions trend (1) \rightarrow (2) \rightarrow (3), and the image from FPV converges on the desired image \mathbf{I}^* .

4.3 Real Flight Experiment

We also deploy our method on a multicopter which features a front-facing camera. We sample the camera position in a dimension of $1.6\text{ m} \times 0.8\text{ m}$ centered on the desired position. The target image for image servoing is a painting. The details of real flight experiments are shown in Fig.4. By sampling with equal spacing, 97 trajectories are captured. Based on the collected data tuple $(\mathbf{r}, \mathbf{u}, \mathbf{I}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathcal{I}$, we first train the network $\psi : \mathcal{I} \rightarrow \mathcal{S}$ to obtain $\mathbf{s}(\mathbf{I}) \in \mathbb{R}^{32}$. Then, we use Algorithm 1 to train to get the controller based on D-learning. Finally, We replace the controller using poses as feedback with a controller using only images as feedback. Experimental results are shown in Fig.5(a). The initial displacement $\Delta \mathbf{r}_0$ is $(-0.60\text{ m}, -0.02\text{ m}, -0.29\text{ m})$. The D-learning controller can stabilize the multicopter using only images as feedback. 3D trajectory pairs based on the PBVS controller using pose as feedback and the D-learning controller using only images as feedback are shown in Fig.5(b).

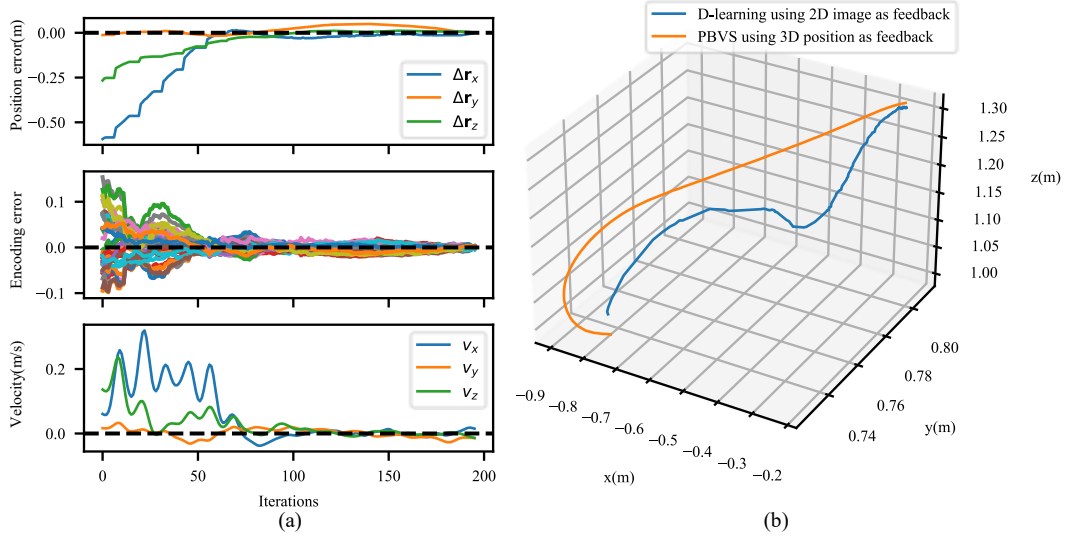


Figure 5: The real flight result for visual servoing. (a) The position error $\Delta \mathbf{r} = \mathbf{r} - \mathbf{r}^*$, the encoding error in the latent space $\mathbf{e} = \mathbf{s}(\mathbf{I}) - \mathbf{s}(\mathbf{I}^*)$, and the velocity of multicopter \mathbf{v} show the effectiveness of CWP based on D-learning. (b) 3D trajectory based on the PBVS controller using 3D position as feedback and the D-learning controller using only 2D image as feedback.

5 Discussion

Conclusion. We propose a sampling-based stability condition, exponential attraction, to meet the Lyapunov stability for learning-based controller. Based on the Theorem 2.1, we propose CWP, which transforms the controller design problem into a pattern classification problem. After that, we propose D-learning for performing CWP without knowledge of the system dynamics. Finally, on the simulation platform and real multicopter platform, we show that our approach can synthesize and verify neural-network controllers for a control system with only images as feedback, and CWP controller has better performance than the controller trained by reinforcement learning.

Limitation. Despite the success in simulated and real flight tasks, our method has not been evaluated in complex practical scenarios. It is anticipated that our approach will prove applicable to more complex real-world robotic control tasks, such as locomotion and navigation. To achieve this goal, our future work needs to further improve data utilization and provide stability guarantees. Moreover, the feature function, Lyapunov function, and controller in the form of neural networks could be learned together in order to achieve superior performance.

References

- [1] R. S. Sutton, and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.
- [2] D. Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.
- [3] N. M. Boffi, S. Tu, N. Matni, J.-J. Slotine, and V. Sindhvani. Learning Stability Certificates from Data. In *Conference on Robot Learning*, pages 1341–1350. PMLR, 2020.
- [4] C. Dawson, S. Gao, and C. Fan. Safe Control With Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction Methods for Robotics and Control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023.
- [5] Y. Chow, O. Nachum, E. A. Duenez-Guzman, and M. Ghavamzadeh. A Lyapunov-based Approach to Safe Reinforcement Learning. *Advances in Neural Information Processing Systems*, 31, 2018.
- [6] M. O’Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S. J. Chung. Neural-fly Enables Rapid Learning for Agile Flight in Strong Winds. *Science Robotics*, 7(66):eabm6597, 2022.
- [7] K. Huang, R. Rana, A. Spitzer, G. Shi, and B. Boots. DATT: Deep Adaptive Trajectory Tracking for Quadrotor Control. *arXiv preprint arXiv:2310.09053*, 2023.
- [8] S. Wang, L. Fengb, X. Zheng, Y. Cao, O. O. Oseni, H. Xu, T. Zhang, and Y. Gao. A Policy Optimization Method Towards Optimal-time Stability. In *Conference on Robot Learning*, pages 1154–1182. PMLR, 2023.
- [9] L. Yang, H. Dai, Z. Shi, C. J. Hsieh, R. Tedrake, and H. Zhang. Lyapunov-stable Neural Control for State and Output Feedback: A Novel Formulation for Efficient Synthesis and Verification. *arXiv preprint arXiv:2404.07956*, 2024.
- [10] Y.-C. Chang, N. Roohi, and S. Gao. Neural Lyapunov Control. *Advances in Neural Information Processing Systems*, 32, 2019.
- [11] C. J. C. H. Watkins, and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
- [12] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake. Lyapunov-stable Neural-network Control. *arXiv preprint arXiv:2109.14152*, 2021.
- [13] R. Zhou, T. Quartz, H. De Sterck, and J. Liu. Neural Lyapunov Control of Unknown Nonlinear Systems with Stability Guarantees. *Advances in Neural Information Processing Systems*, 35, 2022.
- [14] R. O. Duda, E. H. Peter, and P. E. Hart. *Pattern Classification*, John Wiley & Sons, 2006.
- [15] Y. Nesterov. Efficiency of Coordinate Descent Methods on Huge-scale Optimization Problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.
- [16] F. Chaumette, and S. Hutchinson. Visual Servo Control. I. Basic Approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.
- [17] Q. Quan. *Introduction to Multicopter Design and Control*. Springer, 2017.
- [18] S. Felton, E. Fromont, and E. Marchand. Deep Metric Learning for Visual Servoing: when Pose and Image Meet in Latent Space. In *2023 IEEE International Conference on Robotics and Automation*, pages 741–747. IEEE, 2023.
- [19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous Control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971*, 2015.

- 253 [20] A. M. Lyapunov. The General Problem of the Stability of Motion. *International Journal of*
254 *Control*, 55(3):531–534, 1992.
- 255 [21] R. Bellman. Dynamic Programming. *Science*, 153(3731):34–37, 1966.
- 256 [22] E. D. A. Sontag. A ‘Universal’ Construction of Artstein’s Theorem on Nonlinear Stabilization.
257 *Systems & Control Letters*, 13(2):117–123, 1989.
- 258 [23] C. Dawson, Z. Qin, S. Gao, and C. Fan. Safe Nonlinear Control Using Robust Neural
259 Lyapunov-barrier Functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR,
260 2022.
- 261 [24] R. C. B. Rego, and F. M. U. De Araujo. Learning-based Robust Neuro-control: A Method to
262 Compute Control Lyapunov Functions. *International Journal of Robust and Nonlinear Control*,
263 32(5):2644–2661, 2022.
- 264 [25] K. Neumann, A. Lemme, and J. J. Steil. Neural Learning of Stable Dynamical Systems Based
265 on Data-Driven Lyapunov Candidates. In *2013 IEEE/RSJ International Conference on Intelli-*
266 *gent Robots and Systems*, pages 1216–1222. IEEE, 2013.
- 267 [26] H. Ravanbakhsh, and S. Sankaranarayanan. Learning Control Lyapunov Functions from Coun-
268 terexamples and Demonstrations. *Autonomous Robots*, 43:275–307, 2019.
- 269 [27] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames. Episodic Learning with
270 Control Lyapunov Functions for Uncertain Robotic Systems. In *2019 IEEE/RSJ International*
271 *Conference on Intelligent Robots and Systems (IROS)*, pages 6878–6884. IEEE, 2019.
- 272 [28] S. Tesfazgi, A. Lederer, and S. Hirche. Inverse Reinforcement Learning a Control Lyapunov
273 Approach. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 3627–3632.
274 IEEE, 2021.
- 275 [29] F. L. Lewis, and D. Vrabie. Reinforcement Learning and Adaptive Dynamic Programming for
276 Feedback Control. *IEEE Circuits and Systems Magazine*, 9(3):32–50, 2009.
- 277 [30] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In
278 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–
279 778. IEEE, 2016.

A Related Work

Two ways, namely reinforcement learning (RL) [1],[2] and Lyapunov function learning (or certificate learning further, including barrier function and contraction metrics learning) [3],[4], have the potential to handle control problems of complicated systems with big data.

A.1 Reinforcement Learning

As a solution to optimal control problems forward-in-time, RL often focuses on optimization based on the Bellman equation [21]. In the traditional control field, the Bellman equation is often used as an analysis tool rather than a direct design tool in optimization control. However, the hand-design Lyapunov’s method, depending on pseudo-energy functions, is the most popular tool in both analysis and design, aiming to decrease pseudo-energy functions over time so that the state converges to a fixed point. Technically, RL is required to define the rewards function and compute the value function (optimal objectives are defined as priors). In contrast, Lyapunov function learning requires training a parameterized Lyapunov function to match the data set (concrete Lyapunov functions are **NOT** defined as priors). Therefore, they are different in both application and design. RL with Lyapunov functions, where certificates are used to ensure safety or stability, has also been proposed recently [8]. A commonly used certificate is the sum of cost over a limited time horizon as a valid Lyapunov candidate [5]. Lyapunov function learning is more flexible in candidate selection compared to RL.

RL based on the Bellman equation is prevalent in the field of computer science due to its model-free characteristics. More importantly, it can solve very complicated control problems. Compared with RL, Lyapunov’s methods’ achievements on complicated problems with big data are less. Because of the gap between developments by the Lyapunov’s method and the Bellman equation, it is hypothesized that there is an increasing focus on Lyapunov function learning from data. The expectation is to unveil its enormous potential, which is also the major motivation of this paper.

A.2 Lyapunov Function Learning

Lyapunov function learning, which aims to construct Lyapunov functions from data, has two main types of methods:

- Construct a Control Lyapunov Functions (CLF) [22] in formal methods. Lyapunov-stable neural-network control [12], learning-based robust control Lyapunov barrier function [23], neural Lyapunov control [10], and learning-based robust neuro-control [24] employ neural networks to construct both Lyapunov functions and controller simultaneously. These formal methods, that synthesize and verify controllers and Lyapunov function together, formulate the Lyapunov certification problem as proving that certain functions (the Lyapunov function itself, together with the negation of its time derivative) are always non-negative over a domain.
- Learn a certificate in deep learning methods. Demonstration learning [25],[26], episodic Learning [27], and imitation learning [28] aim at only searching for a certificate from given control policy data. Despite the impressive performance of these controllers, many of these controllers require a sufficiently large amount of data to learn semiglobal stabilization, and the data collected from actual robotic systems is expensive.

B Preliminary Remarks

B.1 Exponentially Stability and Exponentially Attraction

In this part, some definitions about stability are given related to the system (1) and the data set (2).

Definition B.1 (Exponentially Stable). For the autonomous system (1), an equilibrium state $\mathbf{x} = \mathbf{0}_{n \times 1}$ is *exponentially stable* if there exist $\alpha, \lambda \in \mathbb{R}_+$ such that $\|\phi(\tau; 0, \mathbf{x}_0)\| \leq \alpha \|\mathbf{x}_0\| e^{-\lambda\tau}$ in

some neighborhoods around the origin. Global exponential stability is independent of the initial state \mathbf{x}_0 .

Here, $\phi(\tau; 0, \mathbf{x}_0)$ represents the solution starting at \mathbf{x}_0 , $\tau \geq 0$. It should be noted that we can only use the data set (2). So, a new definition related to stability, especially for the data set is proposed in the following.

Definition B.2 (Exponentially Attractive on \mathcal{P}). For the autonomous dynamics (1), an equilibrium state $\mathbf{x} = \mathbf{0}_{n \times 1}$ is *exponentially attractive* on the data set \mathcal{P} with $\alpha, \lambda, \varepsilon, \delta \in \mathbb{R}_+$ if $\|\phi(\tau; 0, \mathbf{x})\| \leq \alpha \|\mathbf{x}\| e^{-\lambda\tau}$, $\forall \tau \in [0, \delta]$, $\forall \mathbf{x} \in \mathcal{B}(\mathbf{x}_i, \varepsilon)$ for any $\mathbf{x}_i \in \mathcal{P}$, where $\mathcal{B}(\mathbf{x}_i, \varepsilon)$ denotes a neighborhood around \mathbf{x}_i with radius ε .

Definition B.2 is to describe the trajectory of the autonomous system (1) starting from the state. It is hard or impossible to get the *exponential stability* only based on the data set (2) except for more information on $\mathbf{f}(\mathbf{x})$ obtained further. So, the definition of *exponential attraction* especially for the data set can be served as an intermediate result for classical stability results. For some special systems, we can build the relationship between the *exponential stability* and *exponential attraction*.

Theorem B.3. For the autonomous dynamics $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$, suppose i) the equilibrium state $\mathbf{x} = \mathbf{0}_{n \times 1}$ is exponential attractive on the data set \mathcal{P} with ε , ii) as shown in Fig.6, $\exists l \in \mathbb{R}_+$, $\mathcal{C}_l \subseteq \cup_i \mathcal{B}(\mathbf{x}_i, \varepsilon)$, $\mathbf{x}_i \in \mathcal{P}$, where $\mathcal{C}_l = \{\mathbf{x} \in \mathcal{D} | \mathbf{x}^T \mathbf{P} \mathbf{x} = l\}$ for a positive-definite matrix $\mathbf{0} < \mathbf{P} = \mathbf{P}^T \in \mathbb{R}^{n \times n}$. Then the equilibrium state $\mathbf{x} = \mathbf{0}_{n \times 1}$ is globally exponential stability.

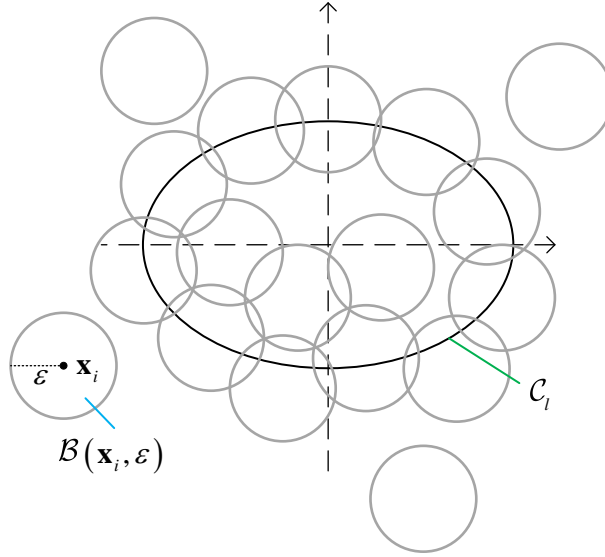


Figure 6: \mathcal{C}_l belongs to the collection of $\mathcal{B}(\mathbf{x}_i, \varepsilon)$.

Proof. For any $\mathbf{x}^* \neq \mathbf{0}_{n \times 1} \in \mathbb{R}^n$, since $\mathbf{x}^{*T} \mathbf{P} \mathbf{x}^* \neq 0$ due to \mathbf{P} being a positive-definite matrix, we have

$$\bar{\mathbf{x}}^* = \theta \mathbf{x}^* \in \mathcal{C}_l$$

where $\theta = \sqrt{\frac{l}{\mathbf{x}^{*T} \mathbf{P} \mathbf{x}^*}}$. Since $\phi(\tau; 0, \mathbf{x}) = e^{\mathbf{A}\tau} \mathbf{x}$, for $\bar{\mathbf{x}}^* \in \mathcal{C}_l$, the solution is $\phi(\tau; 0, \bar{\mathbf{x}}^*)$ satisfying

$$\phi(\tau; 0, \bar{\mathbf{x}}^*) = \theta \phi(\tau; 0, \mathbf{x}^*).$$

On the other hand, since the equilibrium state $\mathbf{x} = \mathbf{0}_{n \times 1}$ is exponential attractive on the data set \mathcal{P} with ε , there exist $\alpha, \lambda, \varepsilon, \delta \in \mathbb{R}_+$ such that $\|\phi(\tau; 0, \bar{\mathbf{x}}^*)\| \leq \alpha \|\bar{\mathbf{x}}^*\| e^{-\lambda\tau}$, $\forall \tau \in [0, \delta]$. Therefore,

$$\begin{aligned} \|\phi(\tau; 0, \mathbf{x}^*)\| &= \frac{1}{\theta} \|\phi(\tau; 0, \bar{\mathbf{x}}^*)\| \\ &\leq \frac{\alpha}{\theta} \|\bar{\mathbf{x}}^*\| e^{-\lambda\tau} \\ &= \alpha \|\mathbf{x}^*\| e^{-\lambda\tau} \end{aligned}$$

$\forall \tau \in [0, \delta]$, $\forall \mathbf{x} \in \mathcal{B}(\mathbf{x}^*, \varepsilon)$ for any $\mathbf{x}_i \in \mathcal{P}$. Therefore, the equilibrium state $\mathbf{x} = \mathbf{0}_{n \times 1}$ is globally exponentially stable. \square

Remark 2. Theorem B.3 implies that, for autonomous linear dynamics, *exponential attraction* is equivalent to *exponential stability* if the data sets cover the boundary of an ellipsoid. For general dynamics, the least amount of data required for the equivalence is worth studying. Some research has applied statistical learning theory to provide probabilistic upper bounds on the generalization error, but these bounds tend to be overly cautious [3].

C Details of Theoretical Analysis

C.1 Assumptions of Theorem 2.1

Assumption C.1. For $\mathbf{x} \in \mathcal{D}$, $\|\mathbf{x}\| \leq d$, where $d \in \mathbb{R}_+$.

Assumption C.2. For $\mathbf{x} \in \mathcal{D}$, the function \mathbf{f} satisfies $\|\partial \mathbf{f}(\mathbf{x}) / \partial \mathbf{x}\| \leq l_f$, where $l_f \in \mathbb{R}_+$.

Assumption C.3. For $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$, there exists $l_g \in \mathbb{R}_+$ such that

$$\left\| \partial \mathbf{g}(\mathbf{x}) / \partial \mathbf{x} \big|_{\mathbf{x}=\mathbf{x}_1} - \partial \mathbf{g}(\mathbf{x}) / \partial \mathbf{x} \big|_{\mathbf{x}=\mathbf{x}_2} \right\| \leq l_g \|\mathbf{x}_1 - \mathbf{x}_2\|.$$

Assumption C.4. For $\mathbf{x} \in \mathcal{D}$, there exist $k_1, k_2 \in \mathbb{R}_+$ such that $k_1 \|\mathbf{x}\|^2 \leq \left\| \mathbf{g}(\mathbf{x})^T \boldsymbol{\theta}_g \right\| \leq k_2 \|\mathbf{x}\|^2$.

Assumption C.5. For $\mathbf{x} \in \mathcal{D}$, there exists a $k_3 \in \mathbb{R}_+$ such that $k_3 \|\mathbf{x}\|^2 < \mathbf{h}(\mathbf{x})^T \boldsymbol{\theta}_h$.

C.2 Proof of Theorem 2.1

Proof. This proof consists of three steps.

Step 1. $\|\Delta \dot{\mathbf{x}}_i\| \leq l_f \|\Delta \mathbf{x}_i\|$. For any $\mathbf{x} \in \mathcal{B}(\mathbf{x}_i, \varepsilon')$, it can be written as

$$\mathbf{x} = \mathbf{x}_i + \Delta \mathbf{x}_i$$

where $\mathbf{x}_i + \Delta \mathbf{x}_i, \mathbf{x}_i \in \mathcal{D}$ and $\Delta \mathbf{x}_i \in \mathcal{B}(\mathbf{0}, \varepsilon')$. Then $\|\Delta \mathbf{x}_i\| \leq \varepsilon'$. In this case, we have

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}_i + \Delta \dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i + \Delta \mathbf{x}_i) \Rightarrow \Delta \dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i + \Delta \mathbf{x}_i) - \mathbf{f}(\mathbf{x}_i).$$

Under Assumption C.2, we further have $\|\Delta \dot{\mathbf{x}}_i\| \leq l_f \|\Delta \mathbf{x}_i\|$.

Step 2. $\|\phi(\tau; 0, \mathbf{x})\| \leq \alpha \|\mathbf{x}\| e^{-\lambda\tau}$ for $\forall \mathbf{x} \in \mathcal{B}(\mathbf{x}_i, \varepsilon)$, where $\varepsilon = \varepsilon' / 2$. For any $\mathbf{x} \in \mathcal{B}(\mathbf{x}_i, \varepsilon')$, according to the definition of $V(\mathbf{x})$ in (3), we have

$$\begin{aligned}
\dot{V}(\mathbf{x}) &= \left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \dot{\mathbf{x}} \right)^T \boldsymbol{\theta}_g \\
&= \left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i+\Delta\mathbf{x}_i} (\dot{\mathbf{x}}_i + \Delta\dot{\mathbf{x}}_i) \right)^T \boldsymbol{\theta}_g \\
&= \left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i + \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i+\Delta\mathbf{x}_i} (\dot{\mathbf{x}}_i + \Delta\dot{\mathbf{x}}_i) - \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i \right)^T \boldsymbol{\theta}_g \\
&= \left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i + \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i+\Delta\mathbf{x}_i} \dot{\mathbf{x}}_i - \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i} \dot{\mathbf{x}}_i + \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i+\Delta\mathbf{x}_i} \Delta\dot{\mathbf{x}}_i \right)^T \boldsymbol{\theta}_g \\
&\leq -\mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\theta}_h + l_g \|\Delta\mathbf{x}_i\| \|\dot{\mathbf{x}}_i\| \|\boldsymbol{\theta}_g\| + l_g \|\mathbf{x}_i + \Delta\mathbf{x}_i\| \|\Delta\dot{\mathbf{x}}_i\| \|\boldsymbol{\theta}_g\| \quad (\text{From Assumption 3}) \\
&\leq -\mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\theta}_h + l_g l_f \|\boldsymbol{\theta}_g\| \|\mathbf{x}_i\| \varepsilon' + l_g l_f \|\boldsymbol{\theta}_g\| (\|\mathbf{x}_i\| + \varepsilon') \varepsilon' \\
&\leq -\mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\theta}_h + 2l_g l_f \|\mathbf{x}_i\| \varepsilon' + l_g l_f \varepsilon'^2.
\end{aligned}$$

368 From Assumption C.5, it is easy to obtain that $-\mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\theta}_h + k_3 \|\mathbf{x}_i\|^2 < 0$, then there exists a
369 $\varepsilon' \in \mathbb{R}_+$ such that $\dot{V}(\mathbf{x}) \leq -k_3 \|\mathbf{x}\|^2$. Then, by Assumption C.4, we have

$$\dot{V}(\mathbf{x}) \leq -k_3 \|\mathbf{x}\|^2 \leq -2\lambda V(\mathbf{x}), \forall \mathbf{x} \in \mathcal{B}(\mathbf{x}_i, \varepsilon')$$

370 where $\lambda = k_3 / 2k_2$. Consequently, for $\mathbf{x} \in \mathcal{B}(\mathbf{x}_i, \varepsilon)$, we have

$$V(\phi(\tau; 0, \mathbf{x})) \leq \|V(\mathbf{x})\| e^{-2\lambda\tau}$$

371 where $\tau \in [0, \tau']$ and τ' is the first time that $\phi(\tau; 0, \mathbf{x})$ is escaping out of $\mathcal{B}(\mathbf{x}_i, \varepsilon')$. Note that
372 $\mathcal{B}(\mathbf{x}_i, \varepsilon) \subset \mathcal{B}(\mathbf{x}_i, \varepsilon')$. As a result, by Assumption C.4, we have

$$\|\phi(\tau; 0, \mathbf{x})\| \leq \alpha \|\mathbf{x}\| e^{-\lambda\tau}$$

373 for $\mathbf{x} \in \mathcal{B}(\mathbf{x}_i, \varepsilon)$, where $\alpha = \sqrt{k_2/k_1}$.

374 Step 3. For any $\mathbf{x} \in \mathcal{B}(\mathbf{x}_i, \varepsilon)$, the escaping time $\tau' > \delta = \frac{\varepsilon}{l_f d}$. Under Assumptions C.1-C.2, since

$$\phi(\tau; 0, \mathbf{x}) - \mathbf{x} = \int_0^\tau \dot{\mathbf{x}}(s) ds$$

375 we have

$$\|\phi(\tau; 0, \mathbf{x}) - \mathbf{x}\| \leq \tau' \|\mathbf{f}(\mathbf{x})\| \leq \tau' l_f d.$$

376 The condition $\|\phi(\tau; 0, \mathbf{x}) - \mathbf{x}\| > \varepsilon = \varepsilon' / 2$ implies that $\phi(\tau; 0, \mathbf{x})$ is escaping out of $\mathcal{B}(\mathbf{x}_i, \varepsilon')$.
377 As a result, we have $\tau' l_f d > \varepsilon$ and then $\tau' > \delta$.

378 With Steps 1-3, there exist $\alpha = \sqrt{k_2/k_1}$, $0 < \lambda < k_3 / 2k_2$, $\varepsilon > 0$, and $\delta = \varepsilon / (l_f d)$ such that
379 $\|\phi(\tau; 0, \mathbf{x})\| \leq \alpha \|\mathbf{x}\| e^{-\lambda\tau}$, $\forall \tau \in [0, \delta]$, $\forall \mathbf{x} \in \mathcal{B}(\mathbf{x}_i, \varepsilon)$. Moreover, $\alpha, \lambda, \varepsilon$ and δ are independent
380 of \mathbf{x}_i , so the result is applicable to any $\mathbf{x}_i \in \mathcal{P}$. Therefore, the equilibrium state $\mathbf{x} = \mathbf{0}_{n \times 1}$ is
381 exponentially attractive on the data set \mathcal{P} .

382 D Details of Feature Extraction

383 To extract the features from the current image \mathbf{I} , we use a ResNet-18 [30] which is trained by metric
384 learning. The main purpose of metric learning is to learn a new metric to reduce the distances
385 between samples of the same class and increase the distances between the samples of different class.

386 In order to better represent the feature, we propose to create a multimodal latent space \mathcal{S} , in which
387 both pose and image representations are mapped. The relationship between latent space and im-
388 ages/poses is illustrated in Fig.7(a). The pose \mathbf{r} maps to a feature embedding \mathbf{s}_r , while the image \mathbf{I}
389 acquired at the pose \mathbf{r} is noted \mathbf{s}_I .

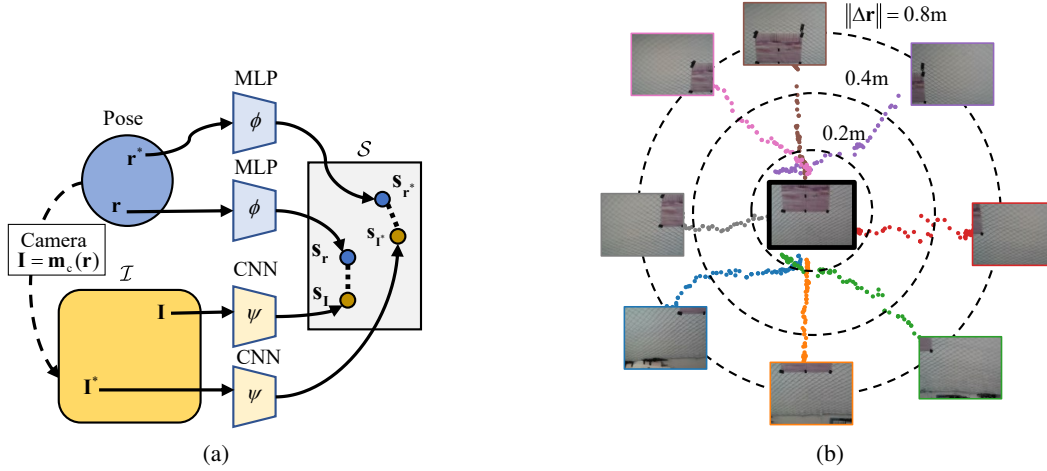


Figure 7: (a) The proposed latent space for visual servoing. Both images and poses are projected in the feature space \mathcal{S} , where they can be compared. (b) PCA projection of trajectories $\mathbf{s}_I - \mathbf{s}_{I^*}$ in the latent space, for 2D motions. Circles show the error for the pose embeddings $\mathbf{s}_r - \mathbf{s}_{r^*}$ for various distances.

We argue that for the best VS behavior, the distance between two embeddings should be equal to the distance between their underlying poses: $d_{\mathcal{S}}(\mathbf{s}_{I_j}, \mathbf{s}_{I_k}) = \|\mathbf{r}_j - \mathbf{r}_k\|_2$, where $d_{\mathcal{S}}$ is the Euclidean distance:

$$d_{\mathcal{S}}(\mathbf{s}_j, \mathbf{s}_k) = \|\mathbf{s}_j - \mathbf{s}_k\|_2. \quad (18)$$

To learn the space \mathcal{S} , we propose to use two distinct, parallel neural networks. The first is $\phi: \mathbb{R}^3 \rightarrow \mathcal{S}$, that maps a pose \mathbf{r} to an embedding $\mathbf{s}_r = \phi(\mathbf{r})$. The second model $\psi: \mathcal{I} \rightarrow \mathcal{S}$, maps an image \mathbf{I} to its latent representation $\mathbf{s}_I = \psi(\mathbf{I})$.

In order to train ϕ and ψ , we devise our loss function $\mathcal{L}_{\mathcal{S}}$ that is based on the distances between latent representations of camera tuple $(\mathbf{r}_j, \mathbf{I}_j)$ and camera tuple $(\mathbf{r}_k, \mathbf{I}_k)$ by

$$\mathcal{L}_{\mathcal{S}} = \mathcal{L}_{\phi, \mathbb{R}^3} + \mathcal{L}_{\psi, \mathbb{R}^3} + \mathcal{L}_{\phi, \psi} \quad (19)$$

where $\mathcal{L}_{\phi, \mathbb{R}^3}$ is the loss function to train $\phi: \mathbb{R}^3 \rightarrow \mathcal{S}$, $\mathcal{L}_{\psi, \mathbb{R}^3}$ is the loss function to train $\psi: \mathcal{I} \rightarrow \mathcal{S}$, and $\mathcal{L}_{\phi, \psi}$ is the loss function to shape the feature space \mathcal{S} in the following

$$\mathcal{L}_{\phi, \mathbb{R}^3} = \text{MSELoss}(\|\mathbf{r}_j - \mathbf{r}_k\|_2, \|\mathbf{s}_{r_j} - \mathbf{s}_{r_k}\|_2) \quad (20a)$$

$$\mathcal{L}_{\psi, \mathbb{R}^3} = \text{MSELoss}(\|\mathbf{s}_{I_j} - \mathbf{s}_{r_k}\|_2, \|\mathbf{s}_{r_j} - \mathbf{s}_{r_k}\|_2) \quad (20b)$$

$$\mathcal{L}_{\phi, \psi} = \text{MSELoss}(\|\mathbf{s}_{I_j} - \mathbf{s}_{I_k}\|_2, \|\mathbf{s}_{r_j} - \mathbf{s}_{r_k}\|_2). \quad (20c)$$

By comparing a representation with every specific tuples, we ensure that a single iteration forces the encoding towards a more stable location. As can be seen in Fig.7(b), the minimization of e in the latent space leads to nearly straight lines in the latent space. The error between pose embeddings also correlates well with the error from image representations.