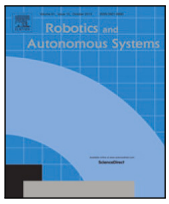
Robotics and Autonomous Systems 190 (2025) 104971

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Imitation learning-based Direct Visual Servoing using the large projection formulation

Sayantan Auddy [a,e,*,1], Antonio Paolillo [b,1], Justus Piater [a,c], Matteo Saveriano [d]

[a] Department of Computer Science, University of Innsbruck, Innsbruck, Austria
[b] Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Lugano, Switzerland
[c] Digital Science Center (DiSC), University of Innsbruck, Innsbruck, Austria
[d] Department of Industrial Engineering, University of Trento, Trento, Italy
[e] Faculty of Electrical Engineering and Computer Science, Technical University of Berlin, Berlin, Germany

ARTICLE INFO

Keywords:
Visual servoing
Learning from demonstration
Learning stable dynamical systems

ABSTRACT

Today robots must be safe, versatile, and user-friendly to operate in unstructured and human-populated environments. Dynamical system-based imitation learning enables robots to perform complex tasks stably and without explicit programming, greatly simplifying their real-world deployment. To exploit the full potential of these systems it is crucial to implement closed loops that use visual feedback. Vision permits to cope with environmental changes, but is complex to handle due to the high dimension of the image space. This study introduces a dynamical system-based imitation learning for direct visual servoing. It leverages off-the-shelf deep learning-based perception modules to extract robust features from the raw input image, and an imitation learning strategy to execute sophisticated robot motions. The learning blocks are integrated using the large projection task priority formulation. As demonstrated through extensive experimental analysis, the proposed method realizes complex tasks with a robotic manipulator.

1. Introduction

Modern robots must be accessible to everyone, as they are rapidly spreading in everyday life environments, from industries [1] to hotels [2] and hospitals [3]. It is expected that a growing number of inexperienced end-users, like children, patients, or elderly people, ask for robots with easy-to-use, friendly, and modular interfaces, endowed with adaptive skills. To meet these requirements, recent advancements in robotics demonstrate the great potential of Machine Learning (Machine Learning (ML)).

From a control perspective, ease of use and adaptability can be obtained with vision-based IL. On one side, IL allows one to easily implement robotic tasks without specific codes [4,5], but by simply following a few demonstrations. Dynamical Systems (DSs) handle the imitation strategy by keeping the stability properties of classical controllers. Such approaches successfully generate complex kinematic motion from previous demonstrations, see [6–9]. On the other, vision-based control like VS [10,11] generates robot behaviors from exteroceptive information, thus taking into account possible changes in the environment. In recent work [12,13], DS-based IL and VS are combined to realize the so-called Imitation Learning Visual Servoing (ILVS). Such a scheme provides dual benefits: (i) additional and complex skills can be imitated (and not explicitly coded) in the VS law; (ii) the use of vision enables adaptive imitation strategies. In this way, the limitations of the original law are overcome by leveraging the information of the demonstrations, e.g., for realizing a visual tracker without an explicit target motion estimator [14].

From a perception point of view, it would be desirable to have modular and transferable blocks that could be easily adapted to the specificity of the deployed robot and the considered task. In this context, DL has been demonstrated to outperform classical computer vision methods [15], also in the robotics domain [16]. Indeed, many approaches based on DL show great performance in detecting and tracking complex objects, e.g., the well-known You Only Look Once (YOLO) algorithm [17,18]. These approaches exhibit robustness, versatility, and even generalization capability, and can solve complex perception tasks in the robotic domain [15,16]. However, sporadic misinterpretations of the raw sensor data, or hallucinations, typical of DL approaches [19], could be disruptive in a closed-loop control scheme.

* Corresponding author.
E-mail addresses: auddy@tu-berlin.de (S. Auddy), antonio.paolillo@idsia.ch (A. Paolillo), justus.piater@uibk.ac.at (J. Piater), matteo.saveriano@unitn.it (M. Saveriano).
[1] These authors contributed equally.

https://doi.org/10.1016/j.robot.2025.104971
Received 13 June 2024; Received in revised form 17 January 2025; Accepted 6 March 2025
Available online 20 March 2025

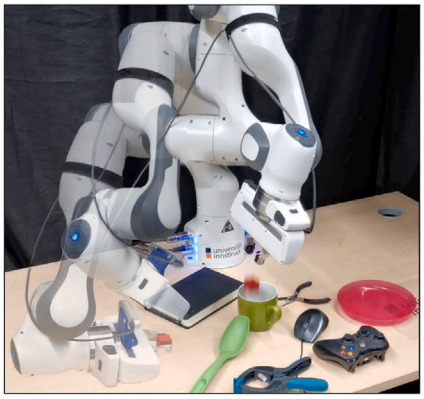


Fig. 1. Our work combines off-the-shelf deep learning strategies to detect objects in the clutter, and imitation learning to realize complicated trajectories, e.g., dropping a cube into a cup on an untidy table. The large projection formulation combines the two machine learning components and ensures convergence to a given target.

Indeed, the measurement accuracy required by a precise control action sets severe requirements, which might be difficult to fulfill by DL-based perception. Specific systems like YOLO, for example, are very reliable in recognizing an object and its position in the camera field of view. However, they fall short in estimating its right orientation, preventing full 6D pose regulation or tracking.

This work aims to realize a robust, modular, stable yet simple VS scheme, taking the good from both data-driven and model-based approaches. We propose a VS architecture that leverages IL and DL paradigms to exploit the potential of state-of-the-art detectors and overcome their limitations in control loops. More in detail, we propose to use an off-the-shelf DL-based detector to obtain a rough but robust visual feedback and refine it by performing IL from a few demonstrations of the desired VS behavior. The learning components are combined in a formal model-based control structure. In this way, we target robotic applications (like dropping a sugar cube in a cup of tea on an untidy table, as exemplified in Fig. 1) proposing an easy solution to complex perception problems, and simple generation of complicated trajectories.

The remainder of the paper is organized as follows. Section 2 discusses the related literature, whereas the technical background of our work is presented in Section 3. Our method and the experimental setup used to validate it are detailed in Sections 4 and 5, respectively. The approach is validated with an extensive experimental analysis, whose results are presented in Section 6. Section 7 concludes the paper with final remarks.

2. Related work

In our vision of adaptive and easy-to-use robots, we must design visual controllers that are straightforward to deploy. In practice, we aim to avoid specific coding to (i) extract the required feedback from dense images and (ii) generate sophisticated trajectories.

Impressive off-the-shelf software releases, e.g., YOLO [17], have been shown to detect objects robustly. It is worth mentioning the large body of work that aims at estimating from vision the target object's pose, see, e.g., [20–23], which can also serve as a control feedback. However, all these approaches implement *standalone* perception systems, i.e., they are unaware of the underlying control structure, and their output might not be accurate enough for control purposes. Specific pose estimators for vision-based control have also been proposed [24–28], but these approaches are sensitive to the operating conditions and usually need intense retraining to operate in different scenarios. Furthermore, pure perception algorithms delegate the generation of sophisticated trajectories to the control block.

One possibility consists of coupling perception and control together in end-to-end learning fashion [29–31]. However, end-to-end methods cannot guarantee stability properties and robustness to disturbances. This is particularly challenging if the robot needs to operate in dynamically changing environments and/or close to the human. Coarse-to-fine imitation learning [32] combines closed- and open-loop execution to perform complex manipulation tasks. One way to ensure stability is to maintain the formal structure of the visual controller, e.g., preserving the VS formalism. In the context of our work, DVS is particularly interesting because it implements VS using direct image measurement, avoiding explicit feature extraction. Examples are VS schemes that use photometric moments [33], pixel luminance [34], histograms [35], or Gaussian mixtures [36] as control feedback. However, in these approaches, the potential of DL is not fully exploited. In [37], an artificial Neural Network (NN) is trained using the knowledge of VS to produce geometrically interpretable features. In [38], an autoencoder is learned to reduce the dimensionality of the image space, and an interaction matrix is directly computed from the network using auto differentiation and utilized in a VS law. Liu et al. [39] simplify the YOLO architecture to speed up object detection, while Luo et al. [40] propose a top-down feature detection network, and both use the predicted features in a VS scheme. These approaches perform well with VS tasks from images without performing classical feature extraction but do not execute complex movements. Our work, instead, provides a unified solution that generates sophisticated robot motion in addition to enhancing the robustness of the perception module.

This paper presents an approach to solve the perception problem and the generation of complex trajectories simultaneously, in the context of vision-based controllers. The proposed framework overcomes the limitations of the literature by proposing an IL-DVS strategy that integrates off-the-shelf DL-based perception modules with IL in a control theoretic framework.

3. Background

Image-based VS [10] is a more than twenty-year-old established technique to regulate a camera to a desired pose through visual information. The most basic law computes 6D velocity commands as $\boldsymbol{v} = -\lambda \hat{\boldsymbol{L}}^{+} \boldsymbol{e}$ [10], zeroing an error $\boldsymbol{e} \in \mathbb{R}^{f}$ defined on the image; λ is a scalar gain; $\hat{\boldsymbol{L}}^{+} \in \mathbb{R}^{f \times 6}$ is the pseudo-inverse of the so-called interaction matrix, relating the camera velocity to the time derivative of the visual feedback. The interaction matrix normally relies on the camera parameters that can be obtained through calibration procedures, and other information like the features' depth that needs to be estimated or approximated. Indeed, the hat over the matrix indicates the approximation due to unknown 3D parameters.

VS can be executed together with other tasks, using the priority scheme established by the null-space projector [41]:

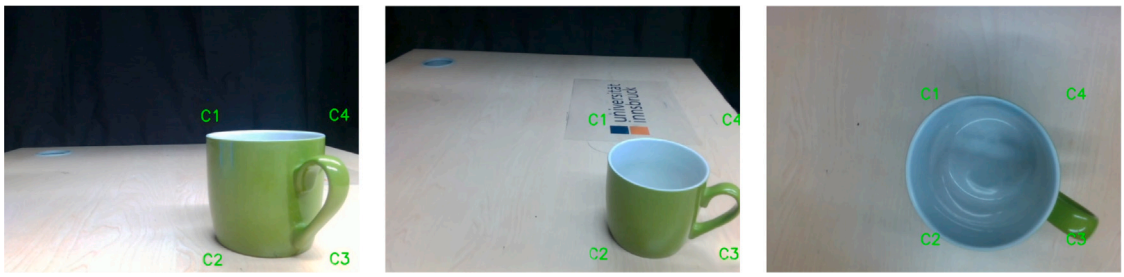
$$\boldsymbol{v}_e = -\lambda \hat{\boldsymbol{L}}^{+} \boldsymbol{e} + \boldsymbol{P} \boldsymbol{\sigma}. \tag{1}$$

The matrix $\boldsymbol{P} = \boldsymbol{I}_6 - \hat{\boldsymbol{L}}^{+} \hat{\boldsymbol{L}}$ is a null-space projector, and $\boldsymbol{\sigma} \in \mathbb{R}^6$ is the desired velocity realizing the secondary task. The main limitation of (1) is that, in normal working conditions, the dimension of the feedback is always greater or equal to the dimension of the task (i.e., $f \geq 6$) [10]. Under these circumstances, there is not much room in the null space of the primary task to execute any other secondary task. Therefore, it has been proposed to use the norm of the error $\eta = \|\boldsymbol{e}\|$ in the primary task [42] and to realize a prioritized control scheme in the following form

$$\boldsymbol{v}_\eta = -\lambda \eta \hat{\boldsymbol{L}}_\eta^{+} + \boldsymbol{P}_\eta \, \boldsymbol{\sigma}, \tag{2}$$

where, similarly to the classic VS scheme (1), $\boldsymbol{P}_\eta$ is a null-space projector and $\boldsymbol{\sigma} \in \mathbb{R}^6$ is the desired velocity realizing the secondary task. The matrices of interest can be retrieved in closed form [42], and are reported here for convenience:

$$\hat{\boldsymbol{L}}_\eta^{+} = \eta \frac{\hat{\boldsymbol{L}}^{\top} \boldsymbol{e}}{\boldsymbol{e}^{\top} \hat{\boldsymbol{L}} \hat{\boldsymbol{L}}^{\top} \boldsymbol{e}} \quad \text{and} \quad \boldsymbol{P}_\eta = \boldsymbol{I}_6 - \frac{\hat{\boldsymbol{L}}^{\top} \boldsymbol{e} \, \boldsymbol{e}^{\top} \hat{\boldsymbol{L}}}{\boldsymbol{e}^{\top} \hat{\boldsymbol{L}} \hat{\boldsymbol{L}}^{\top} \boldsymbol{e}}.$$



(a) Side view. (b) Oblique view. (c) Top view.

Fig. 2. State-of-the-art DL-based systems like YOLO can be used to detect the features of an object of interest on a raw monocular image robustly. Examples of features are the vertices (denoted with 'C1', 'C2', 'C3', and 'C4') of the bounding box detected around the image of a cup. However, such detection systems fail to capture the correct object orientation. In the three snapshots, YOLO provides very similar feature values that correspond to three very different relative camera–object orientations producing a side (a), oblique (b), and top view (c) of the cup. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Note that the control law (2) drives the system towards the original desired behavior, as $\boldsymbol{e} \to \boldsymbol{0}$ for $\eta \to 0$. The operator $\boldsymbol{P}_\eta$ is called *large projector* as the law (2) ensures enough room in the null space to achieve secondary tasks. In practice, regulating the scalar norm instead of the vector error releases degrees of freedom during the transient for other secondary tasks. Using simple Lyapunov arguments, it is possible to show that the law (2) is locally stable and that the secondary task does not impact the stability. If the secondary velocities $\boldsymbol{\sigma}$ are not compatible with the primary task, they are simply ignored and not executed, as the effect of the construction of the projector operator [41]. In (2) the presence of singularities requires the use of a switching strategy around $\eta = 0$:

$$\boldsymbol{v} = \alpha(\eta)\, \boldsymbol{v}_\eta + \big(1 - \alpha(\eta)\big)\, \boldsymbol{v}_e \tag{3}$$

where α is a scalar variable smoothly changing from 1 to 0 in the vicinity of $\eta = 0$, allowing a switch to the classical law (1) from (2) [42].

This control system has been recently used to implement a stable IL [13] to realize complex VS tasks. In particular, the main task error η stably drives the system towards steady-state convergence; the secondary task is used to imitate demonstrated velocities $\boldsymbol{\sigma}$ during the transient. In this work, we use such a control structure to handle the output of a state-of-the-art DL-based feature detector and overcome its limitations by leveraging the information of a few demonstrations, using the IL paradigm, as detailed in the following section.

4. Approach

Our objective is to exploit the control redundancy offered by the large projector formalism (2) to integrate DL and IL for direct VS. We exploit the great potential of state-of-the-art DL-based object detectors, which are treated as an underlying raw detector. Furthermore, our framework uses IL to look at previous demonstrations to overcome the limitations of DL-based detectors. The scheme further exploits IL to realize complex trajectories. In this section, we explain in detail the DL and IL learning components, and how they are combined using the large null-space projector control structure. More in detail, we first present and formulate the perception problem in Section 4.1, which regards the limitation of the used detectors in the control context; then, we present our solution to the problem introducing the imitation strategy (in Section 4.2) and the whole control scheme (Section 4.3) of our approach.

4.1. DL-based detection and its limits

The DL-based detector is a pre-trained NN that is fed with raw camera images and provides as output a measure of the object in the form of visual features

$$\boldsymbol{f} = \boldsymbol{m}_\theta(\boldsymbol{i}), \tag{4}$$

where $\boldsymbol{i} \in \mathbb{R}^{3wh}$ is a vectorized colored image with a size of $w \times h$ pixels, $\boldsymbol{m}$ is the detector model, and $\boldsymbol{\theta} \in \mathbb{R}^p$ denotes the parameters of the pre-trained model; the output $\boldsymbol{f} \in \mathbb{R}^m$ contains features of the detected objects, such as the corners of its bounding box measured on the image (see Fig. 2). Following the classic VS rationale, such features are compared with a set of desired values, denoted with $\boldsymbol{f}^*$, to provide a measure of the visual error, from which the control action evolves. Indeed, the desired set of features is obtained by the model fed with a reference desired image $\boldsymbol{i}^*$, i.e., $\boldsymbol{f}^* = \boldsymbol{m}_\theta(\boldsymbol{i}^*)$. Therefore, the visual error to be considered in the standard VS law (1) is computed as

$$\boldsymbol{e} = \boldsymbol{f} - \boldsymbol{f}^*, \tag{5}$$

whereas its norm, to be used in the large projection formulation (2), is

$$\eta = \|\boldsymbol{f} - \boldsymbol{f}^*\| \tag{6}$$

where $\|\cdot\|$ denotes the Euclidean norm.

State-of-the-art systems, such as the YOLO algorithm [17] considered in our work, can perform high-frequency and robust feature detection. However, such features are not truly informative of the real pose of the observed object and, thus, not enough for full servoing of the camera pose. Typically, the bounding boxes detected by YOLO are rectangles centered on the image of the object of interest and aligned to the image borders. Such bounding boxes do not include any information about the object orientation, as shown in Fig. 2. An extended version of YOLO, YOLOv8 [43], computes bounding boxes that are oriented in the plane of the image. However, even in this case, the detected bounding box features do not contain information about the complete orientation of the detected object in all three rotation axes. Thus, they are insufficient for controlling the full 6D pose of the camera and its motion in the Cartesian space. One possible solution would be to refine the model $\boldsymbol{m}_\theta$ by fine-tuning the parameters $\boldsymbol{\theta}$. However, this solution requires engineering work and the intervention of specialized scientists. Furthermore, the achievement of satisfactory results remains challenging. Our solution, instead, proposes to learn how to overcome these limitations of the off-the-shelf DL detection module from given demonstrations of the desired task.

4.2. Overcoming the detection limits through imitation

To overcome the limitations of DL-based detectors like YOLO, we leverage the information contained in a set of human-demonstrated trajectories. In particular, the corrective action for servoing the 3D orientation can be imitated from demonstrations of the full VS behavior. Such demonstrations are contained in a dataset with this shape:

$$\mathcal{D} = \left\{ \boldsymbol{f}_t^n, \boldsymbol{p}_t^n, \boldsymbol{r}_t^n \right\}_{t,n}^{T,N}, \tag{7}$$

where the subscripts t and n denote the tth sample of the nth demonstration, respectively; N is the total number of demonstrations and

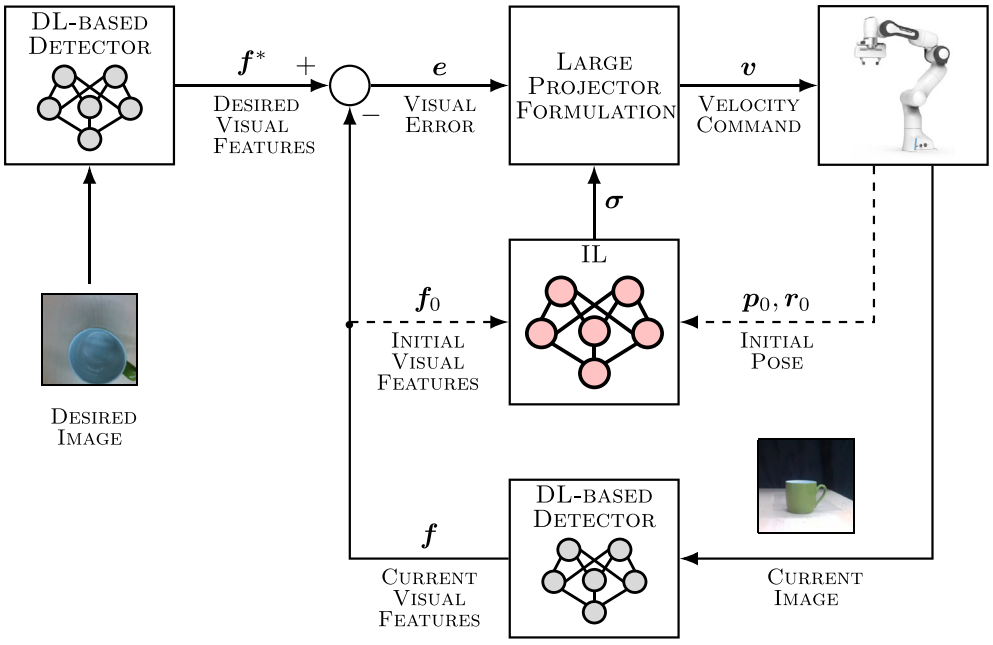


Fig. 3. The proposed framework for IL-DVS exploits a detection model (a frozen DL network, implemented by YOLO) to extract features from raw images robustly and IL (implemented as a fine-tuned NODE network) to realize complex trajectories and overcome the limitation of the detection model. The large projection formulation merges the output of the detection and imitation strategy in a closed-loop control law resulting in accurate and converging robot movements.

T is the length (expressed as the total number of samples) of each demonstration; $\boldsymbol{f}$ indicates the visual features vector (as defined in Section 4.1), and $\boldsymbol{p} \in \mathbb{R}^3$ is the robot's end-effector's position. The end-effector orientation $\boldsymbol{r} \in \mathbb{R}^3$ is obtained by projecting a unit quaternion $\boldsymbol{q} \in \mathbb{S}^3$ into the *tangent space* placed at the goal quaternion using the so-called *logarithmic map* [44]. Such quantities are obtained by showing the full desired behavior to the robot by using, e.g., teleoperation or kinesthetic teaching. During the collection of demonstrations, the object of interest is placed at a fixed location w.r.t. the robot, and the object is always maintained in the camera field of view.

The IL strategy is realized by augmenting the detection model with additional layers. More in detail, we augment the detector architecture with a Neural Ordinary Differential Equation (NODE) solver [45] used for IL. The additional NODE layers are trained on the dataset $\mathcal{D}$. NODE has previously been used for IL [46] because it can be trained easily with only a few demonstrations, is extremely fast during inference, and exhibits accurate empirical performance for real-world full 6 degrees-of-freedom trajectory learning tasks [47]. The lack of mathematical stability guarantees of the trajectories predicted by a stand-alone NODE is addressed in our approach by the large projector formalism that guarantees that the position trajectory of the robot will not diverge, as discussed in Section 5. Hence, we do not use other alternatives such as [48] that assure stability but have a much slower inference speed [46].

NODE assumes that the training data are instances of a nonlinear dynamical system that maps a generic input state $\boldsymbol{x}$ into an output that consists of its time derivative $\dot{\boldsymbol{x}}$. In our setting, the state at time sample t is $\boldsymbol{x}_t = \left(\boldsymbol{f}_t^\top, \boldsymbol{p}_t^\top, \boldsymbol{r}_t^\top\right)^\top$. To accurately approximate the underlying dynamics, NODE optimizes a set of parameters ϑ by minimizing a sum-of-square-error loss. It is worth mentioning that, having projected unit quaternions in the tangent space, we can readily use the dataset $\mathcal{D}$ as in (7) to train NODE. This is a common strategy in robotics [44,49–51], which is also effective for NODE [46]. After training NODE, the rotation component is transformed back into unit quaternions using the so-called *exponential map* [44,47]. While training NODE, in each iteration we extract from the N demonstrations in $\mathcal{D}$ a short contiguous segment of length T_s, obtained by drawing from $\mathcal{D}$ elements at random temporal locations T_s, $T_s \ll T$ [46]. We then concatenate each element of $\mathcal{D}_s$ into the vectors $\boldsymbol{x}_t^n = \left(\boldsymbol{f}_t^{n\top}, \boldsymbol{p}_t^{n\top}, \boldsymbol{r}_t^{n\top}\right)^\top$, $t = 1, \dots, T_s$, $n = 1, \dots, N$. Given the input vectors $\boldsymbol{x}_t^n, \forall t, n$, NODE uses its internal neural network $\boldsymbol{n}_\vartheta$ (called *target network*) to produce derivatives of the input that are then numerically integrated to produce a predicted trajectory $\hat{\boldsymbol{x}}_t^n, \forall t, n$. For training NODE, we used the mean squared error loss $\mathcal{L}$, defined as:

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\sum_{n=1}^{N}\sum_{t=1}^{T_s} \|\boldsymbol{x}_t^n - \hat{\boldsymbol{x}}_t^n\|_2^2 \\ &= \frac{1}{2}\sum_{n=1}^{N}\sum_{t=1}^{T_s} \|\boldsymbol{f}_t^n - \hat{\boldsymbol{f}}_t^n\|_2^2 + \|\boldsymbol{p}_t^n - \hat{\boldsymbol{p}}_t^n\|_2^2 + \|\boldsymbol{r}_t^n - \hat{\boldsymbol{r}}_t^n\|_2^2 .\end{aligned} \tag{8}$$

Ultimately, the trained NODE's target network is the following model

$$\boldsymbol{\sigma} = \boldsymbol{n}_\vartheta\left(\hat{\boldsymbol{f}}, \hat{\boldsymbol{p}}, \hat{\boldsymbol{r}}\right) \tag{9}$$

that is initialized with the initial state of the system comprised of the output $\boldsymbol{f}_0$ of the pre-trained detection model (i.e., YOLO) and the initial robot's position $\boldsymbol{p}_0$ and tangent space orientation $\boldsymbol{r}_0$. As output, it produces the robot velocity $\boldsymbol{\sigma} \in \mathbb{R}^6$, which is the time derivative of $\boldsymbol{p}$ and $\boldsymbol{r}$, imitating the complex trajectories demonstrated in the dataset. In subsequent steps of the robot's motion, NODE evolves in an open-loop fashion its internal belief of the current state of $\hat{\boldsymbol{f}}$, $\hat{\boldsymbol{p}}$, and $\hat{\boldsymbol{r}}$, while keeping on predicting the velocity $\boldsymbol{\sigma}$.

4.3. Merging DL and IL with the large projector

The DL-based detection model and the NODE target network are deployed within the robot control loop, as shown in the schematic of our approach (Fig. 3). Given the desired and current image, the DL-based detector extracts the visual features, as in (4), which are then used to compute the visual error $\boldsymbol{e}$ and its norm η. These values are needed to compute the primary task in (3). The lower priority (secondary) task considers the corrective velocity $\boldsymbol{\sigma}$ as regressed from the NODE target network (9).

In our scheme, the pre-trained YOLO model represents the DL-based detector. The higher-priority task uses the current features (detected from the camera image) to adapt to changes in the object's location. This feedback term also ensures convergence to the desired visual features $\boldsymbol{f}^*$ (i.e., $\boldsymbol{e} \to 0$ for $t \to +\infty$). At the same time, the NODE

network realizes an open-loop IL strategy that lets the robot execute more complex motions without affecting the convergence.

5. Experimental setup

In this section, we describe in detail our setup, including the hardware and software systems used in our experiments. We describe how we collect demonstrations, train NODE, and specify the metrics used for evaluation.

5.1. Hardware and software components

We use a Franka Emika Panda robot, a 7-degrees-of-freedom robotic arm. It features advanced force sensing and collision detection capabilities, making it safe and suitable for manipulation tasks in collaborative environments. The robot is fixed on a tabletop and equipped with an Intel RealSense D435 camera at the end effector. We run our image detection pipeline on a computer with an Intel i5-7640X CPU, 32 GB RAM, and an NVIDIA GeForce RTX 4060 Ti GPU. The robot control software runs on a separate computer with a real-time OS kernel on the same network.

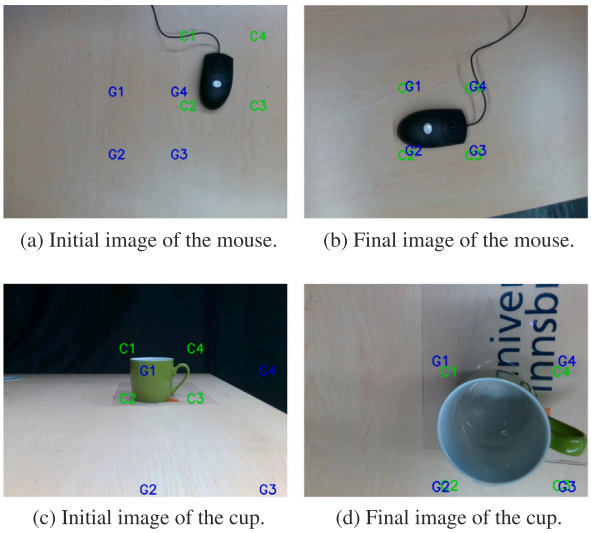
Our software is implemented using ROS noetic and we use the `franka_example_controllers`[2] package to communicate with the robot. The ROS interface accepts pose-level input, which we obtain by integrating the velocity command, computed as in (2) Furthermore, we experimentally verified that position and orientation tasks are decoupled in (2). This is because the features from YOLO (obtained from the squared bounding box) do not contain information on the orientation; recall Fig. 2. Therefore, as a difference from the general block diagram in Fig. 3, the velocity output of NODE is split into its linear and angular parts; the first actually enters the large projection formulation, whereas the latter goes directly to the robot. This implementation detail implies that the control structure maintains the robot stability in position, while the execution of the orientation is delegated to the mere imitation strategy. Nevertheless, it is worth mentioning that in our experiments, we observe that, with few demonstrations, the robot can perform safe behaviors even in orientation.

We use the `realsense2_camera`[3] package to communicate with the camera and run YOLO with the `darknet_ros`[4] package, capturing RGB camera images with a size of 640 × 480 pixels at a framerate of 30 Hz. We use the standard calibration procedures provided by the `cv2`[5] Python library to determine the camera's intrinsic parameters (consisting of the focal length and the central point). Instead, to determine the extrinsic camera parameters (i.e., the pose of the camera with respect to the robot's gripper), we use the `aruco_ros`[6] package and the `cv2` library. NODE is implemented and trained in PyTorch. Our open-source code, including the necessary software dependencies and calibration scripts, is available at https://github.com/sayantanauddy/il-dvs.

5.2. YOLO detector

The YOLO detector in our setup (see Fig. 3) uses the `yolov2-tiny` [18] model pretrained on the COCO dataset [52]. The underlying YOLO network can be easily changed to any of the other pre-trained models provided by the `darknet_ros`[4] package. In our experiments, we select the object of interest (e.g., the mouse or the cup) from the list of all objects detected by YOLO and use the detected features of this object for VS.

[2] https://wiki.ros.org/franka_example_controllers.
[3] https://wiki.ros.org/realsense2_camera.
[4] http://wiki.ros.org/darknet_ros.
[5] https://opencv.org/.
[6] https://wiki.ros.org/aruco_ros.



(a) Initial image of the mouse. (b) Final image of the mouse.

(c) Initial image of the cup. (d) Final image of the cup.

Fig. 4. Initial (left) and final images (right) captured by the robot camera in the experiments with the mouse (top) and the cup (bottom). Desired visual features are shown in blue and denoted with the letter "G", whereas the current visual features are the green letters "C". (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The rectangular bounding boxes originally predicted by YOLO often show significant variations in aspect ratio and vertex positions, even in consecutive frames that appear visually identical. These spurious changes cause abrupt jumps in the computed visual error that can disrupt closed-loop control dynamics and impede target convergence. This issue is particularly critical near the end of the robot's trajectory, where the current bounding box nearly overlaps with the target bounding box, potentially leading to convergence problems. Therefore, we convert the rectangular bounding boxes into squares with side lengths equal to the larger dimension of the original rectangle while maintaining the same center point. To further mitigate noise in the detected visual features (i.e., vertices of the bounding box), we apply average filtering over the past 50 frames. As the object of interest is not subject to rapid movements in the camera frame, smoothing the error signal facilitates convergence without compromising the robot's speed, as demonstrated in the supplementary video.[7]

We use the vertices of the resulting bounding box as features in our VS scheme. In the presentation of our results, such features are denoted with "C_i" whereas their desired counterparts are "G_i", with $i = 1, \ldots, 4$.

5.3. Collection of demonstrations

For training NODE, we collect demonstrations via kinesthetic teaching [5]. The object under consideration is placed in a specific location, and a human user physically guides the robot's end-effector from an initial pose to the desired final pose. We collect two sets of demonstrations corresponding to two different objects. In the first set, a computer mouse is placed on the table with the robot's camera looking down at the mouse; kinesthetic demonstrations are provided so that the image of the mouse is rotated 90° clockwise in the final pose of the robot. The latter set of demonstrations is collected using a cup. In the initial pose for these demonstrations, the camera looks side-on at a cup on the table; in the final one, the robot's end-effector is positioned on top of the cup with the camera looking down. See Fig. 4 for a visual reference of the initial and final poses for the mouse and cup demonstrations. Each set contains four demonstrations used to train a NODE (one for each set). Each demonstration consists of a sequence of 500 steps.

[7] https://youtu.be/b0lviYlXarI.

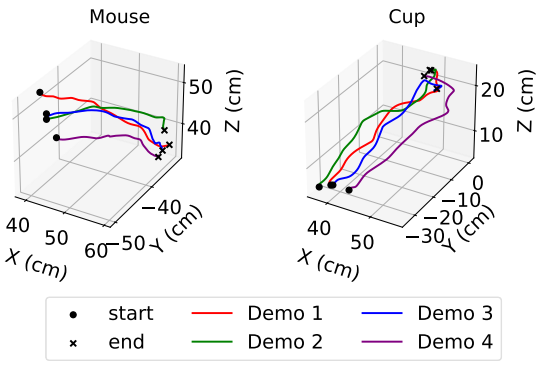


Fig. 5. Position trajectories of demonstrations provided for the "Centering the mouse in the image" (left) and "Dropping an object in the cup" (right) tasks. Diversity is introduced by starting from different initial poses and also through the differences between each kinesthetic demonstration.

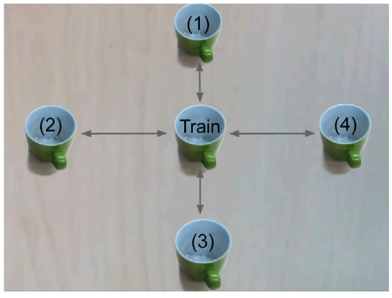


Fig. 6. The object position used during the collection of the demonstrations is the one at the center, whereas the four novel object positions used for the evaluation are 15 cm off the center.

In each set, the object's position remains unchanged but diversity is introduced into the collected data by starting each demonstration from a different pose. Furthermore, manual kinesthetic demonstrations result in different trajectories and also introduce diversity into the training data. The position trajectories of the demonstrations collected for the two tasks are depicted in Fig. 5.

5.4. NODE training

The NODE target network predicts the derivatives of the inputs, and during training, numerical integration is used to generate trajectories from the predictions [46]. We use a NODE target network with 2 hidden layers containing 256 neurons each and ReLU activations. In each demonstration n and each step t of the training data, the inputs and outputs of the NODE are 10-dimensional, consisting of the upper left and lower right vertices of the bounding box (i.e., the visual features) $\hat{\boldsymbol{f}}_t^n \in \mathbb{R}^4$ in image coordinates normalized to lie within $[0.0, 100.0]$, the position (in cm) of the robot's end-effector in the task space $\boldsymbol{p}_t^n \in \mathbb{R}^3$, and the rotation vector $\boldsymbol{r}_t^n \in \mathbb{R}^3$ obtained by projecting the orientation quaternions of the end-effector to the local tangent space, as described in Section 4.2. Following [46], we scale the rotation vectors by a constant factor of 100.0 so that all input features are of comparable magnitudes. For each recorded demonstration set (mouse and cup), we train a NODE for 2×10^4 iterations with a learning rate of 5×10^{-4} using the loss defined in (8).

Note that the sides of the bounding boxes detected by YOLO are always parallel to the image sides (the image coordinates of only the upper left and lower right vertices of the bounding boxes are predicted). Consequently, the visual features that are recorded to train the NODE are also 4-dimensional. Our VS scheme uses a general representation of a bounding box consisting of the features corresponding to all 4 vertices. Therefore, during inference, we compute the 8-dimensional visual features by deriving the coordinates of the upper right and lower left vertices from the YOLO predictions.

5.5. Evaluation protocol

Our analysis compares the performance of three VS schemes. The first is denoted with IIL and uses a NODE instance trained on the demonstrations to control the robot in an end-to-end fashion. The second one is a classic VS scheme where YOLO provides the required visual feedback; following the literature [34,35], we call it DVS, as it is a *direct* approach considering the whole image as input. Finally, our proposed method, augmenting the ILVS scheme [12] with DL-based direct measurement, is called IL-DVS. It is worth mentioning that, for a fair comparison, all the approaches are fed with the square and filtered bounding boxes computed as discussed in Section 5.2.

We conduct separate experiments for the mouse and the cup. For each experiment, we evaluate the performance of the three schemes for five different object positions: one as in the demonstrations, and four unseen positions, as shown in Fig. 6. We run each test for $T = 700$ time steps, where T is the demonstration length, and measure the norm of the visual error. and the end-effector position and orientation error at the final step of the experiment.

All errors are computed at the final step of the robot's trajectory. In particular, the visual error norm η is computed using (6). The end-effector position error is computed as

$$\delta = \|\boldsymbol{p}_T - \boldsymbol{p}_g\|_2 \tag{10}$$

where $\boldsymbol{p}_g \in \mathbb{R}^3$ is the Cartesian position of the robot at the last step of the demonstration (i.e., the ground truth desired position) and $\boldsymbol{p}_T \in \mathbb{R}^3$ is the final position reached during the evaluation. The orientation error is computed using the orientation $\boldsymbol{q}_T \in \mathbb{S}^3$ of the robot in the last step of the experiment and $\boldsymbol{q}_g \in \mathbb{S}^3$ the orientation at the end of the demonstration (i.e., the ground truth desired orientation). The measure of the orientation error is computed as

$$\varepsilon = \left\| \log\left(\boldsymbol{q}_T \otimes \bar{\boldsymbol{q}}_g\right) \right\|_2 \tag{11}$$

where $\log(\cdot)$ denotes the logarithmic map [44], $\bar{\boldsymbol{q}}_g$ denotes the conjugation of quaternion $\boldsymbol{q}_g$, whereas the symbol '$\otimes$' is the quaternion product operator. Note that the position error δ is not computed for novel object positions as there is no ground truth end-effector position $\boldsymbol{p}_g$ for these object positions. However, the relative orientation of the robot's end-effector with respect to the object at the end of execution is expected to be the same irrespective of whether the object is placed at the demonstrated or novel locations. This enables us to measure the orientation error ε for trained as well as novel object positions.

Furthermore, for the cup experiment, we measure the success of dropping a small object into the cup at the end of the robot's motion. Since resetting the robot after every evaluation may introduce some stochasticity, we evaluate each method on each object position three times. Finally, we also perform a qualitative evaluation of our IL-DVS scheme in a cluttered scene where an object is to be dropped into a cup at different positions among several other objects.

6. Experimental results

During the evaluation of the different approaches described in Section 5.5, the same trained NODE is used in our IL-DVS approach as well as in IIL where NODE controls the robot in an end-to-end way. This section presents the results of the different approaches in the mouse and cup experiments. Examples of the presented experiments can be viewed in the supplementary video.[8]

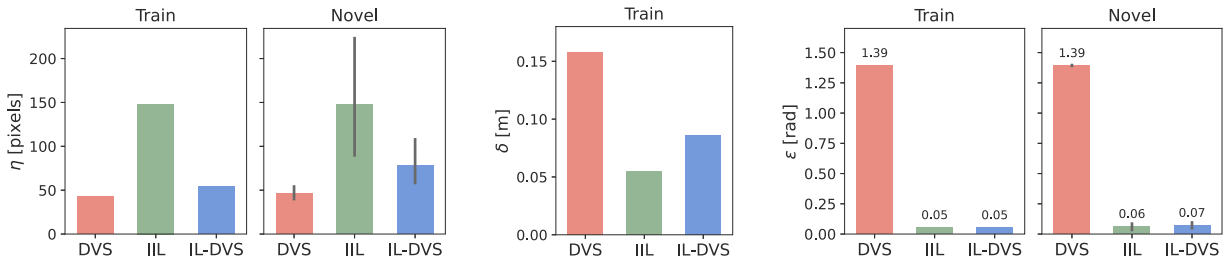


Fig. 7. Centering the mouse in the image: the average norm of the final visual error (left), position (center), and orientation error (right) achieved with the three schemes, starting from similar trained positions or novel ones. All errors are computed at the final step of the robot's trajectory. Note that the end-effector position error cannot be computed for the novel positions. Colored boxes show the means and error bars show the 95% confidence interval. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

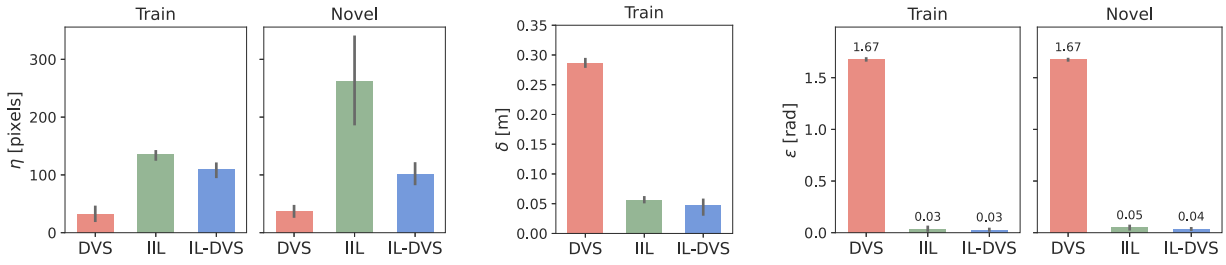
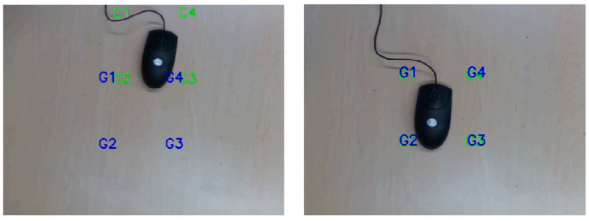


Fig. 8. Dropping an object in the cup: the average norm of the final visual error (left), position (center), and orientation error (right) achieved with the three schemes, starting from similar trained positions or novel ones. All errors are computed at the final step of the robot's trajectory. Note that the end-effector position error cannot be computed for the novel positions. Colored boxes show the means and error bars show the 95% confidence interval. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Initial image. (b) Final image with DVS.

(c) Final image with IIL. (d) Final image with IL-DVS.

Fig. 9. Centering the mouse in the image with novel mouse locations: initial and final images reached with the different methods. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.1. Centering the mouse in the image

The goal of the first set of experiments is to move the robot such that the image of the mouse (whose visual features are shown in green in Fig. 9) coincides with its desired image (corresponding to a desired set of features, shown in blue in Fig. 9). Note that the task also requests the robot camera (and end-effector) to rotate in order to reach the desired relative camera–mouse pose. We evaluate each of the three schemes (DVS, IIL, IL-DVS) on five different object positions, as described in Section 5.5, and report the overall results in Fig. 7. DVS achieves low visual errors but fails to control the orientation correctly. IIL achieves low orientation errors but cannot adapt to novel object positions leading to a high visual error. In contrast, our IL-DVS approach exhibits a low visual error, can adapt to novel object positions, and controls the robot orientation properly.

More in detail, DVS achieves the lowest visual error as VS with YOLO alone can position the robot camera such that the current and desired visual features match very closely. However, as the features detected by YOLO do not have any information about the real orientation of the mouse, the yaw orientation of the robot's end-effector does not change at all. As a result, the final orientation reached in this experiment is very different from the desired one (compare the desired pose in Fig. 4 with the final pose achieved by DVS in Fig. 9(b)). The IIL approach can achieve low orientation error as the underlying NODE has been trained to achieve the demonstrated orientation. However, it cannot compensate for the changes in the novel mouse positions outside the demonstrations, resulting in high visual error. This effect can also be qualitatively observed in Fig. 9(c). Our IL-DVS approach can adapt to novel positions of the mouse. At the same time, it utilizes the trained NODE to achieve the correct orientation as shown by the kinesthetic demonstrations. This enables IL-DVS to take advantage of both DVS and IL and achieve low visual and orientation errors, making it the best approach among the ones we evaluate. Fig. 9(d) shows that IL-DVS achieves a close fit to the desired visual features and the desired orientation.

6.2. Dropping an object in the cup

The second set of experiments aims to drive the robot end-effector on top of the cup and drop an object in it, leveraging the demonstrations recorded with the cup. Note that these experiments present the additional challenge of performing nontrivial (e.g., nonlinear) trajectories. The camera's initial and desired views are representative of two completely different camera–cup relative poses, as shown in Fig. 4.

The quantitative evaluation is presented in Fig. 8. The different approaches are again evaluated for one trained object position and

[8] https://youtu.be/b0lviYlXarI.

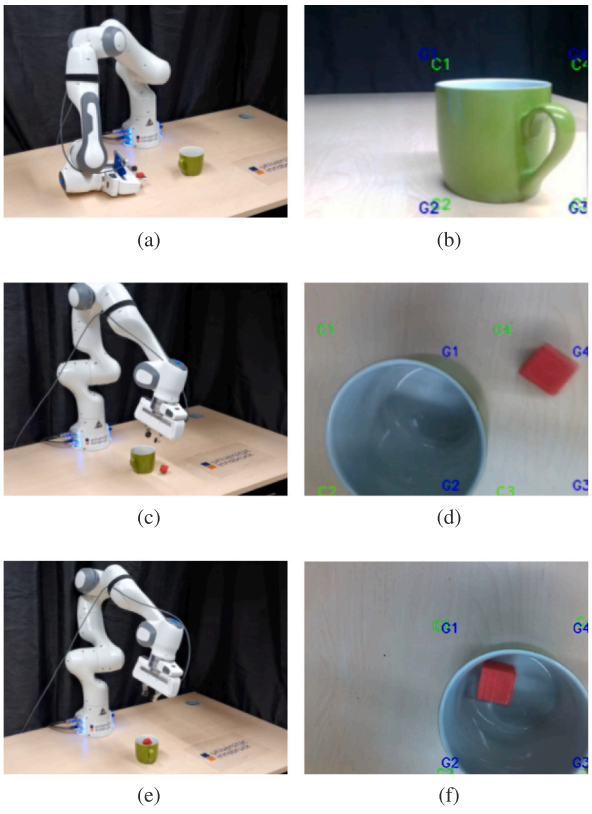


Fig. 10. Dropping an object in the cup placed in a novel position: robot external view (left column) and the corresponding camera view (right column) as executed by DVS (a,b), IIL (c,d) and IL-DVS (e,f).

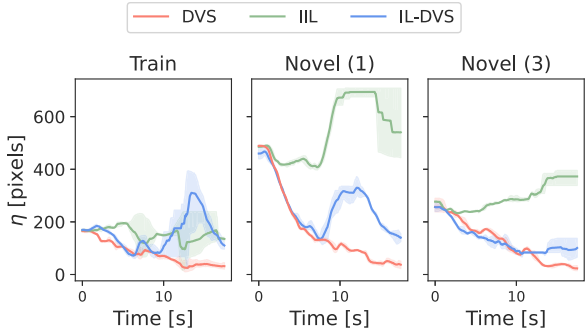


Fig. 11. Drop an object in the cup: the visual error with the trained object position (left) and two novel positions (center and right).

four novel object positions outside the demonstrations, as described in Section 5.5. DVS achieves a low visual error but fails to move the end-effector above the cup and orient it properly, resulting in high position and orientation errors. IIL achieves a low orientation error but cannot adapt to novel object positions leading to a high visual error. As expected, the visual error is much higher for the novel object positions; the orientation error remains low for both trained and novel object positions. IL-DVS, i.e., our approach, achieves low errors for both position and orientation (corresponding to limited visual errors) as it can adapt to novel object positions and orient the robot properly.

A qualitative evaluation is shown in Fig. 10 and confirms the quantitative results. The DVS approach can align the visual features to their desired counterpart (Fig. 10(b)), but the robot ends up in a completely wrong pose (Fig. 10(a)). As a result, it cannot drop the grasped object into the cup. IIL attempts to realize the complex trajectory required to execute the dropping task, but it has poor accuracy (Fig. 10(c)).

Table 1
Success rates for dropping an object into the cup.

Approach	Success rate [%]		
	Train	Novel	Overall
DVS	0.00	0.00	0.00
IIL	83.33	0.00	16.67
IL-DVS	**100.00**	**95.83**	**96.67**

In particular, for a novel object position, as the one shown in the figure, the robot is unable to adapt and drops the object outside the cup (Fig. 10(d)). IL-DVS (ours) shows the best performance among all the approaches, as it can cope with changing positions of the cup and drops the object successfully, as shown in Figs. 10(e) and 10(f).

Fig. 11 shows the time evolution of the visual error for the trained object position and two novel object positions during a complete experiment (see Fig. 6 for a description of the trained and novel object positions). IL-DVS (ours) achieves a low visual error for both novel as well as trained object positions, whereas IIL achieves much higher visual errors for novel object positions due to its inability to adapt. As expected, the visual error made by DVS stays low throughout.

We also report the success rates of dropping the object into the cup for all methods (see Table 1). Each approach is evaluated 15 times as described in Section 5.5 (5 object positions for each of the 3 trials per object position) and we report mean values for success. A trial is considered 100% successful if the object drops cleanly into the cup, 50% successful if the object hits the cup's rim but eventually falls inside, and 0% otherwise. Table 1 shows that our IL-DVS approach achieves near-perfect results, while IIL achieves a much lower score since it is unsuccessful in dropping the object into the cup placed at novel positions; DVS is never able to drop the object into the cup and gets a score of 0.

6.3. Handling cluttered scenes

We execute the experiments presented in the previous section in a cluttered setting and qualitatively evaluate the effectiveness of our IL-DVS approach. The cup is placed on the table among several other objects, such as a book, a plate, a clamp, a spatula, and a game controller. In different trials, the location of the cup is varied among the other objects (see Fig. 12).

The pre-trained YOLO object detection model identifies multiple objects in the scene, as shown in Fig. 13 (left). As YOLO provides a list of detected object names and their corresponding visual features, we can easily select the object of interest (the cup, in our case) and use its visual features for VS, see Fig. 13 (right). Once the cup is selected as the desired object, the robot executes the required motion to position its gripper over the cup. Additionally, in a cluttered scene, YOLO offers us the flexibility of easily changing the target object to any other object detected in the scene.

With our IL-DVS approach, the robot successfully drops the object into the cup in a cluttered setting and also adapts its pose to the different locations of the cup, as shown in Fig. 12 (note that the locations of the cup and the corresponding final poses of the robot in these snapshots). Finally, Fig. 14 shows some views of the object being dropped into the cup placed in different cluttered scenes, as seen from the robot's end-effector camera.

7. Conclusion

In this paper, we have presented Imitation Learning-based Direct Visual Servoing (IL-DVS), a dynamical system-based imitation learning approach for direct visual servoing. The proposed framework overcomes several limitations of existing approaches. IL-DVS exploits off-the-shelf deep learning-based perception to extract features from raw camera images, augmented with imitation learning layers that generate

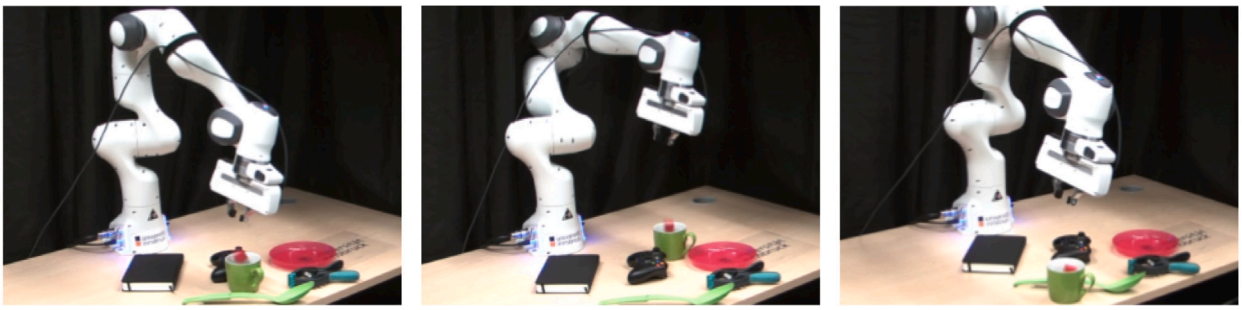


Fig. 12. With IL-DVS, the robot successfully drops the object into the cup placed in different positions of a cluttered table.

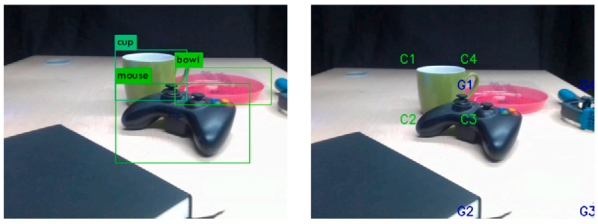


Fig. 13. YOLO detects a cup robustly even in a cluttered scene (left), providing the required visual features shown with green letters 'C' (right); the desired features are also shown, with blue letters 'G'. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 14. Two final image frames captured by the robot camera showing that IL-DVS successfully drives the robot to drop the object into the cup in different cluttered environments.

complex robot trajectories. A key difference from end-to-end learning approaches is that IL-DVS exploits a control theoretical framework to ensure convergence to a given target. The approach has been extensively evaluated with real robot experiments, and compared with two baselines showing superior performance.

CRediT authorship contribution statement

Sayantan Auddy: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Antonio Paolillo:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Justus Piater:** Supervision, Project administration. **Matteo Saveriano:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Funded by the European Union projects INVERSE (grant agreement No. 101136067) and SERMAS (grant agreement No. 101070351), and by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 22.00247.

Sayantan Auddy was supported by a Doctoral Scholarship from the University of Innsbruck's Support Programme for Young Researchers, awarded by the University of Innsbruck, Vice-Rectorate for Research.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2025.104971.

Data availability

The link to our open source git repository containing our code and data has been shared in the paper.

References

[1] A. Grau, M. Indri, L.L. Bello, T. Sauter, Robots in industry: The past, present, and future of a growing collaboration with humans, IEEE Ind. Electron. Mag. 15 (1) (2020) 50–61.

[2] M.M.O. Youngjoon Choi, S.S. Kim, Service robots in hotels: understanding the service quality perceptions of human-robot interaction, J. Hosp. Mark. Manag. 29 (6) (2020) 613–635.

[3] C.S. González-González, V. Violant-Holz, R.M. Gil-Iranzo, Social robots in hospitals: a systematic review, Appl. Sci. 11 (13) (2021) 5976.

[4] B.D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robot. Auton. Syst. 57 (5) (2009) 469–483.

[5] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Robot programming by demonstration, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer, 2008, pp. 1371–1394.

[6] S.M. Khansari-Zadeh, A. Billard, Learning stable non-linear dynamical systems with Gaussian mixture models, IEEE Trans. Robot. 27 (5) (2011) 943–957.

[7] S.M. Khansari-Zadeh, A. Billard, Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions, Robot. Auton. Syst. 62 (6) (2014) 752–765.

[8] N. Perrin, P. Schlehuber-Caissier, Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems, Systems Control Lett. 96 (2016) 51–59.

[9] J. Urain, M. Ginesi, D. Tateo, J. Peters, ImitationFlow: Learning deep stable stochastic dynamic systems by normalizing flows, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020, pp. 5231–5237.

[10] F. Chaumette, S. Hutchinson, Visual servo control. Part I: Basic approaches, IEEE Robot. Autom. Mag. 13 (4) (2006) 82–90.

[11] F. Chaumette, S. Hutchinson, Visual servo control. Part II: Advanced approaches, IEEE Robot. Autom. Mag. 14 (1) (2007) 109–118.

[12] A. Paolillo, M. Saveriano, Learning stable dynamical systems for visual servoing, in: IEEE International Conference on Robotics and Automation, 2022, pp. 8636–8642.

[13] A. Paolillo, P. Robuffo Giordano, M. Saveriano, Dynamical system-based imitation learning for visual servoing using the large projection formulation, in: IEEE International Conference on Robotics and Automation, 2023, pp. 755–761.

[14] R. Felici, M. Saveriano, L. Roveda, A. Paolillo, Imitation learning-based visual servoing for tracking moving objects, in: International Workshop on Human-Friendly Robotics, Springer, 2023, pp. 110–122.
[15] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G.V. Hernandez, L. Krpalkova, D. Riordan, J. Walsh, Deep learning vs. traditional computer vision, in: Computer Vision Conference, 2020, pp. 128–144.
[16] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[17] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[18] P. Jiang, D. Ergu, F. Liu, Y. Cai, B. Ma, A review of Yolo algorithm developments, Procedia Comput. Sci. 199 (2022) 1066–1073.
[19] P. Sahoo, P. Meharia, A. Ghosh, S. Saha, V. Jain, A. Chadha, Unveiling hallucination in text, image, video, and audio foundation models: A comprehensive survey, 2024, arXiv:2405.09589.
[20] Y. Li, G. Wang, X. Ji, Y. Xiang, D. Fox, Deepim: Deep iterative matching for 6D pose estimation, in: European Conference on Computer Vision, 2018, pp. 683–698.
[21] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, D. Fox, Self-supervised 6D object pose estimation for robot manipulation, in: IEEE International Conference on Robotics and Automation, 2020, pp. 3665–3671.
[22] M. Nava, A. Paolillo, J. Guzzi, L.M. Gambardella, A. Giusti, Uncertainty-aware self-supervised learning of spatial perception tasks, IEEE Robot. Autom. Lett. 6 (4) (2021) 6693–6700.
[23] M. Nava, A. Paolillo, J. Guzzi, L.M. Gambardella, A. Giusti, Learning visual localization of a quadrotor using its noise as self-supervision, IEEE Robot. Autom. Lett. 7 (2) (2022) 2218–2225.
[24] A. Saxena, H. Pandya, G. Kumar, A. Gaud, K.M. Krishna, Exploring convolutional networks for end-to-end visual servoing, in: IEEE International Conference on Robotics and Automation, 2017, pp. 3817–3823.
[25] C. Yu, Z. Cai, H. Pham, Q.-C. Pham, Siamese convolutional neural network for sub-millimeter-accurate camera pose estimation and visual servoing, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2019, pp. 935–941.
[26] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, P. Corke, Training deep neural networks for visual servoing, in: IEEE International Conference on Robotics and Automation, 2018, pp. 3307–3314.
[27] P. Durdevic, D. Ortiz-Arroyo, A deep neural network sensor for visual servoing in 3d spaces, Sensors 20 (5) (2020) 1437.
[28] P. Vitiello, K. Dreczkowski, E. Johns, One-shot imitation learning: A pose estimation perspective, in: Conference on Robot Learning, 2023, pp. 943–970.
[29] S. Felton, E. Fromont, E. Marchand, Siame-se(3): regression in se(3) for end-to-end visual servoing, in: IEEE International Conference on Robotics and Automation, 2021, pp. 14454–14460.
[30] E.Y. Puang, K. Peng Tee, W. Jing, KOVIS: Keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020, pp. 7527–7533.
[31] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (1) (2016) 1334–1373.
[32] E. Johns, Coarse-to-fine imitation learning: Robot manipulation from a single demonstration, in: IEEE International Conference on Robotics and Automation, 2021, pp. 4613–4619.
[33] M. Bakthavatchalam, O. Tahri, F. Chaumette, A direct dense visual servoing approach using photometric moments, IEEE Trans. Robot. 34 (5) (2018) 1226–1239.
[34] C. Collewet, E. Marchand, Photometric visual servoing, IEEE Trans. Robot. 27 (4) (2011) 828–834.
[35] Q. Bateux, E. Marchand, Histograms-based visual servoing, IEEE Robot. Autom. Lett. 2 (1) (2017) 80–87.
[36] N. Crombez, E.M. Mouaddib, G. Caron, F. Chaumette, Visual servoing with photometric Gaussian mixtures as dense features, IEEE Trans. Robot. 35 (1) (2019) 49–63.
[37] A. Paolillo, M. Nava, D. Piga, A. Giusti, Visual servoing with geometrically interpretable neural perception, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2022, pp. 5300–5306.
[38] S. Felton, P. Brault, E. Fromont, E. Marchand, Visual servoing in autoencoder latent space, IEEE Robot. Autom. Lett. 7 (2) (2022) 3234–3241.
[39] H. Liu, D. Li, B. Jiang, J. Zhou, T. Wei, X. Yao, MGBM-YOLO: a faster lightweight object detection model for robotic grasping of bolster spring based on image-based visual servoing, J. Intell. Robot. Syst. 104 (4) (2022) 77.
[40] J. Luo, L. Zhu, L. Li, P. Hong, Robot visual servoing grasping based on top-down keypoint detection network, IEEE Trans. Instrum. Meas. 73 (2024) 1–11.
[41] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modelling, planning and control, Springer, 2009.
[42] M. Marey, F. Chaumette, A new large projection operator for the redundancy framework, in: IEEE International Conference on Robotics and Automation, 2010, pp. 3727–3732.
[43] G. Jocher, A. Chaurasia, J. Qiu, Ultralytics YOLO, 2023, URL https://github.com/ultralytics/ultralytics.
[44] A. Ude, B. Nemec, T. Petrić, J. Morimoto, Orientation in cartesian space dynamic movement primitives, in: 2014 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2014, pp. 2997–3004.
[45] R.T. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 6572–6583.
[46] S. Auddy, J. Hollenstein, M. Saveriano, A. Rodríguez-Sánchez, J. Piater, Continual learning from demonstration of robotics skills, Robot. Auton. Syst. 165 (2023) 104427.
[47] S. Auddy, J. Hollenstein, M. Saveriano, A. Rodríguez-Sánchez, J. Piater, Scalable and efficient continual learning from demonstration via hypernetwork-generated stable dynamics model, 2024, arXiv preprint arXiv:2311.03600.
[48] J. Urain, M. Ginesi, D. Tateo, J. Peters, Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 5231–5237.
[49] Y. Huang, F.J. Abu-Dakka, J. Silvério, D.G. Caldwell, Toward orientation learning and adaptation in cartesian space, IEEE Trans. Robot. 37 (1) (2020) 82–98.
[50] M. Saveriano, F. Franzel, D. Lee, Merging position and orientation motion primitives, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 7041–7047.
[51] W. Wang, M. Saveriano, F.J. Abu-Dakka, Learning deep robotic skills on Riemannian manifolds, IEEE Access 10 (2022) 114143–114152.
[52] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft COCO: Common objects in context, in: European Conference Computer Vision, Springer, 2014, pp. 740–755.

Sayantan Auddy is a researcher in the Learning and Intelligent Systems lab at the Technical University of Berlin (Germany). He received his PhD degree from the University of Innsbruck (Austria) in 2025. He received his M.Sc. degree in Intelligent Adaptive Systems from the University of Hamburg (Germany) in 2018, and his B.Tech. degree in Computer Science and Engineering from Haldia Institute of Technology (India) in 2009. His research interests include continual learning for robotics, reinforcement learning, and deep learning.

Antonio Paolillo is a Researcher at the Dalle Molle Institute for Artificial Intelligence (IDSIA, USI-SUPSI) in Lugano. He received his Ph.D. and M.Sc. from Sapienza University of Rome, Italy, in 2015 and 2011, respectively. He was a postdoc at CNRS-University of Montpellier, France (2015–17); Idiap Research Institute, Martigny, Switzerland (2018–19); EPFL, Lausanne, Switzerland (2019–20). He visited Örebro University, Sweden (2010); CNRS-University of Montpellier, France (2014); and CNRS-AIST Joint Robotics Laboratory, Tsukuba, Japan (2015). He is an Associate Editor for RA-L. His research interests include robotic control, machine learning and AI for robotics, human–robot interaction, and rehabilitation robotics.

Justus Piater is a professor of computer science at the University of Innsbruck, Austria, where he leads the Intelligent and Interactive Systems group and serves as the founding director of the interdisciplinary Digital Science Center. He earned his Ph.D. degree at the University of Massachusetts Amherst, USA, and was a professor of computer science at the University of Liège, Belgium, and a visiting researcher at the Max Planck Institute for Biological Cybernetics in Tübingen, Germany. His research interests focus on learning and inference in sensorimotor systems. He has published more than 200 papers, several of which have received best-paper awards.

Matteo Saveriano received his B.Sc. and M.Sc. degree in automatic control engineering from University of Naples, Italy, in 2008 and 2011, respectively. He received his Ph.D. from the Technical University of Munich in 2017. Currently, he is an assistant professor at the Department of Industrial Engineering (DII), University of Trento, Italy. Previously, he was an assistant professor at the University of Innsbruck and a postdoctoral researcher at the German Aerospace Center (DLR). He is an Associate Editor for RA-L. His research activities include robot learning, human–robot interaction, and understanding and interpreting human activities. Webpage: https://matteosaveriano.weebly.com/