DURableVS: Data-efficient Unsupervised Recalibrating Visual Servoing via online learning in a structured generative model

Nishad Gothoskar[1] Miguel Lázaro-Gredilla[2]
Yasemin Bekiroglu[2] Abhishek Agarwal[2]
Joshua B. Tenenbaum[1] Vikash K. Mansinghka[1] Dileep George[2]

***Abstract*—Visual servoing enables robotic systems to perform accurate closed-loop control, which is required in many applications. However, existing methods either require precise calibration of the robot kinematic model and cameras or use neural architectures that require large amounts of data to train. In this work, we present a method for unsupervised learning of visual servoing that does not require any prior calibration and is extremely data-efficient. Our key insight is that visual servoing does not depend on identifying the veridical kinematic and camera parameters, but instead only on an accurate generative model of image feature observations from the joint positions of the robot. We demonstrate that with our model architecture and learning algorithm, we can consistently learn accurate models from less than 50 training samples (which amounts to less than 1 min of unsupervised data collection), and that such data-efficient learning is not possible with standard neural architectures. Further, we show that by using the generative model in the loop and learning online, we can enable a robotic system to recover from calibration errors and to detect and quickly adapt to possibly unexpected changes in the robot-camera system (e.g. bumped camera, new objects).**

I. Introduction

In robotics, techniques that use visual feedback to guide robot control are referred to as visual servoing [5], [20], [36]. But while visual servoing enables closed-loop control, it often relies on calibration of the robot-camera system [38]. In particular, it requires precise calibration of the cameras observing the robot and the kinematic model of the robot itself [26]. Calibration procedures may require human input or precisely manufactured devices and are often tedious to perform. Recent work has attempt to circumvent the need for calibration by instead learning visual servoing, often using neural network architectures [24], [35]. However, these approaches require large amounts of data and time to train in order to achieve good performance. Further, they cannot quickly adapt to changes in the robot-camera system (e.g. bumped camera, new objects) without retraining.

In this paper, we present a learning-based approach to visual servoing that, rather than employing neural networks, uses structured generative models. We propose a differentiable model architecture and optimization procedure for learning the generative model's parameters from data. While our architecture resembles standard kinematic and camera modeling, a key insight of our work is that we can perform visual servoing without identifying the veridical (real world) kinematic and

[1]Massachusetts Institute of Technology [2]Vicarious AI

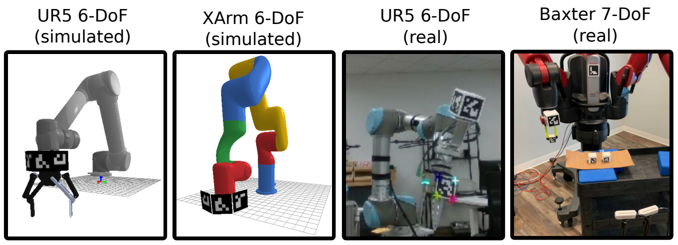


Fig. 1. Evaluation environments. We evaluated DurableVS in 4 different environments: two simulated environments with a UR5 (6-DoF) and Xarm (6-DoF) and two real environments with a UR5 and a Baxter (7-DoF).

TABLE I
DurableVS versus VISP [26] versus neural method

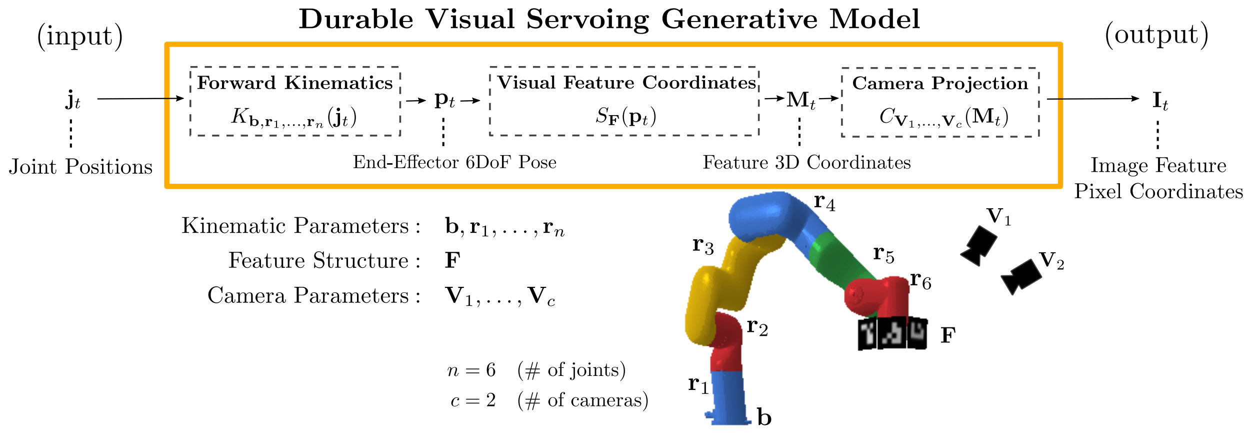
Method	Data-Efficient	Unsupervised	Re-calibrating
VISP [26]	✔	requires calibration	
Neural		✔	
Ours	✔	✔	✔

camera parameters, but rather just using an accurate generative model of image feature coordinates from joint positions. Our experimental results demonstrate that we can learn accurate models extremely data-efficiently, from <50 training samples (<1 min of unsupervised data collection), and that such data-efficient learning is not possible with standard neural architectures. Further, since our model is fast enough to run in the control loop and learning can be done online, our method can be used to detect and adapt to unexpected changes in the robot-camera system.

In the remainder of this paper, we introduce our model architecture, learning algorithm, and inference procedure. Then we present experimental results highlighting our approach's accuracy, data-efficiency, speed, and its ability to quickly recover from changes to the robot-camera setup via online learning. We conclude by discussing related work in vision-based control, limitations, and future work.

II. DurableVS Model

Our architecture is composed of three modules: (1) Forward Kinematics, (2) Visual Feature Structure, and (3) Camera Projection. The *Forward Kinematics* module takes as input the joint positions $\mathbf{j}_t$ and predicts the 6DoF Cartesian pose $\mathbf{p}_t$ of the end-effector. The *Visual Feature Structure* module takes



Durable Visual Servoing Generative Model
(input)
(output)
$\mathbf{j}_t$
Joint Positions
Forward Kinematics
$K_{\mathbf{b},\mathbf{r}_1,\ldots,\mathbf{r}_n}(\mathbf{j}_t)$
$\mathbf{p}_t$
End-Effector 6DoF Pose
Visual Feature Coordinates
$S_{\mathbf{F}}(\mathbf{p}_t)$
$\mathbf{M}_t$
Feature 3D Coordinates
Camera Projection
$C_{\mathbf{V}_1,\ldots,\mathbf{V}_c}(\mathbf{M}_t)$
$\mathbf{I}_t$
Image Feature Pixel Coordinates
Kinematic Parameters : $\mathbf{b},\mathbf{r}_1,\ldots,\mathbf{r}_n$
Feature Structure : $\mathbf{F}$
Camera Parameters : $\mathbf{V}_1,\ldots,\mathbf{V}_c$
$n=6$ (# of joints)
$c=2$ (# of cameras)
$\mathbf{r}_1$ $\mathbf{r}_2$ $\mathbf{r}_3$ $\mathbf{r}_4$ $\mathbf{r}_5$ $\mathbf{r}_6$ $\mathbf{b}$ $\mathbf{F}$ $\mathbf{V}_1$ $\mathbf{V}_2$

Fig. 2. We illustrate the 3 modules that are composed to define our generative model from joint positions $\mathbf{j}_t$ to image observations $\mathbf{I}_t$, and the parameters of those modules. According to the kinematic structure of the robot, the joint positions $\mathbf{j}_t$ will place the robot's end-effector at a certain 6D Cartesian pose $\mathbf{p}_t$. Given the robot end-effector's pose and the relative coordinates of the visual features in the end-effector frame, we can compute the 3D coordinates $\mathbf{M}_t$ of the features in the world frame. Finally, the features are observed through the cameras and appear at pixel coordinates $\mathbf{I}_t$ in the images.

as input the 6DoF Cartesian pose $\mathbf{p}_t$ of the end-effector and predicts the 3D coordinates $\mathbf{M}_t$ of the visual features that are rigidly attached to the end-effector. The *Camera Projection* module takes as input the 3D coordinates $\mathbf{M}_t$ of the visual features and predicts the pixel coordinates $\mathbf{I}_t$ at which the visual features appear in the observed images. We compose these 3 modules, in sequence, to get a model that, given the joint positions $\mathbf{j}_t$ as input, predicts the pixel coordinates $\mathbf{I}_t$. This architecture parallels the causal generative process by which placing a robot at certain joint positions results in visual features attached to the end-effector being observed at certain locations in the images. (See Figure 2). We now discuss the specific parametrization of each of the 3 modules:

Forward Kinematics We parametrize the Forward Kinematics module using the Denavit–Hartenberg (DH) convention [14]. For each of the n links of an n-DoF robot, there are 4 DH parameters: ω - joint angle offset; d - link offset; a - link length; α - link twist. We collect these parameters per link into vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n$. The link parameters $\mathbf{r}_i$ and the current joint angle $\mathbf{j}_t[i]$ together define a relative transformation $L_{\mathbf{r_i}}(\mathbf{j}_t[i])$ between consecutive links. Composing these relative transformation for all links in the kinematic chain gives the pose of the end-effector in the robot's base frame. We additionally parametrize the 6DoF pose of the base frame as $\mathbf{b}$. The Forward Kinematics K takes as input the joint positions $\mathbf{j}_t$ and outputs the 6DoF Cartesian end-effector pose $\mathbf{p}_t$:

$$\mathbf{p}_t = K_{\mathbf{b},\mathbf{r}_1,\ldots,\mathbf{r}_n}(\mathbf{j}_t) = \mathbf{b} \cdot \prod_{i=1}^{n} L_{\mathbf{r}_i}(\mathbf{j}_t[i]) \tag{1}$$

The parameters that must be learned are: $\mathbf{b}, \mathbf{r}_1, \ldots, \mathbf{r}_n$. We refer to them collectively as $\mathbf{R}$.

Visual Feature Structure We assume that we can track visual features that are rigidly attached to the robot's end-effector. Thus, these features can be described by their 3D (relative) coordinates $\mathbf{F}$ in the end-effector's reference frame. The Visual Feature Structure module S takes as input the 6DoF Cartesian pose $\mathbf{p}_t$ of the end-effector in the world frame and outputs the 3D coordinates $\mathbf{M}_t$ of the visual features in the world frame, done via a homogeneous transformation:

$$\mathbf{M}_t = S_{\mathbf{F}}(\mathbf{p}_t) = \mathbf{p}_t \cdot \mathbf{F}^\top \tag{2}$$

The parameter that must be learned is: $\mathbf{F}$.

Camera Projection We model cameras using a pinhole camera model whose parameters are the extrinsics matrix $\mathbf{E}$, which defines the relative pose of the camera in the world frame, and the intrinsic matrix $\mathbf{K}$, which governs how 3D points in the camera frame are projected onto the image plane via focal lengths and principal points [37]. Each of the c cameras has its own parameters and we use $\mathbf{K}_i$ and $\mathbf{E}_i$ to denote the intrinsics and extrinsics of camera i, respectively. The Camera Projection C takes as input the 3D homogeneous coordinates $\mathbf{M}_t$ of the features and outputs the 2D coordinates of those features in the image $\mathbf{I}_t^{(i)}$ as observed in camera i:

$$\mathbf{I}_t^{(i)} = C_{\mathbf{K}_i,\mathbf{E}_i}(\mathbf{M}_t) = \mathbf{K} \cdot \mathbf{E} \cdot \mathbf{M}_t \tag{3}$$

The parameters that must be learned are: $\mathbf{K}_i$ and $\mathbf{E}_i$ for each camera $i = 1, \ldots, c$ where c is the number of cameras. We collect the intrinsics and extrinsics per camera and refer to them as $\mathbf{V}_i$ for camera i. To simplify notation, when we call C on the input $\mathbf{M}_t$ with all camera parameters $\mathbf{V}_1, \ldots, \mathbf{V}_c$, we assume the output is the predicted image features $\mathbf{I}_t^{(1)}, \ldots, \mathbf{I}_t^{(c)}$ in all c cameras, which we collect and refer to as $\mathbf{I}_t$.

Summary Our architecture composes the 3 modules described above:

$$\begin{aligned}
\text{Forward Kinematics:}\quad & \mathbf{p}_t &&= K_{\mathbf{R}}(\mathbf{j}_t)\\
\text{Visual Feature Structure:}\quad & \mathbf{M}_t &&= S_{\mathbf{F}}(\mathbf{p}_t)\\
\text{Camera Projection:}\quad & \mathbf{I}_t &&= C_{\mathbf{V_1},\ldots,\mathbf{V_c}}(\mathbf{M}_t)\\
\textbf{Full Architecture:}\quad & \mathbf{I}_t &&= C_{\mathbf{V_1},\ldots,\mathbf{V_c}}(S_{\mathbf{F}}(K_{\mathbf{R}}(\mathbf{j}_t)))
\end{aligned}$$

Importantly, each of the above modules is differentiable and thus the full architecture is differentiable.

III. Learning

We learn the parameters of our model from training data consisting of pairs $(\mathbf{j}_t, \mathbf{I}_t)$ of joint positions and corresponding image feature coordinates. This data can be collected in a completely unsupervised manner by sending random actions to the robot and capturing images from the cameras. Given this data, we learn the parameters of the model via stochastic gradient-based optimization with L-BFGS [25] optimizing the reconstruction error between the predictions of the model (given $\mathbf{j}_t$ as input) and the true observations $\mathbf{I}_t$.

While the ground truth veridical parameters (i.e. the true robot kinematic model, feature structure, and camera intrinsics and extrinsics) represent one solution to this optimization problem, it is not the only solution. Consider parameter settings corresponding to scaled up, scaled down, rotated, and translated version of the real world. These models would be equally accurate at predicting image observations from joint positions, even though the model's latent 3D representation may not necessarily coincide with the real world. A key insight of our work is that these models can be used for visual servoing and can be learned unsupervised.

In the remainder of this section, we describe the 3 subroutines of our learning algorithm:

A. Camera and Feature Structure Learning

We first optimize the camera parameters $\mathbf{V}_{1:c}$ (for all c cameras) and feature coordinate parameters $\mathbf{F}$. In addition to optimizing these parameters, we also optimize the end-effector poses $\mathbf{p}_{1:T}$, which are a variable in the model. In this first stage of the optimization, we are considering a truncated model from poses $\mathbf{p}_t$ to pixel coordinates $\mathbf{I}_t$, and ignoring the joint positions and kinematics. The function D_{pixel} is a quadratic error between the observed and generated pixel coordinates. The objective of the following optimization is to find a setting of the parameters ($\mathbf{V}_{1:c}$,$\mathbf{F}$) and poses ($\mathbf{p}_{1:T}$) that produces pixel coordinates consistent with the actual observed coordinates $\mathbf{I}_t$ (for all c cameras and T time steps):

$$\mathbf{F}^*,\ \mathbf{V}^*_{1:c},\ \mathbf{p}^*_{1:T} = \underset{\mathbf{F},\mathbf{V}_{1:c},\mathbf{p}_{1:T}}{\operatorname{argmin}} \sum_{t=1}^{T}\sum_{i=1}^{c} D_{\text{pixel}}\big(C_{\mathbf{V}_i}(S_{\mathbf{F}}(\ \mathbf{p}_t)), \mathbf{I}_t^{(i)}\big) \tag{4}$$

$\mathbf{I}_t^{(i)}$ are the feature detections at time t in camera i. This optimization is prone to convergence to sub-optimal local minima and it is important to provide good initialization for $\mathbf{p}_{1:T}$ and $\mathbf{V}_{1:c}$. We compute initializations by triangulating features observed in multiple cameras and estimating the camera baselines i.e the transforms between pairs of cameras.

B. Kinematic Learning

Next, we optimize the kinematic parameters $\mathbf{R}$. We use the observed joint positions $\mathbf{j}_{1:T}$ and the Cartesian end-effector poses $\mathbf{p}_{1:T}$ output from the previous subroutine. The function $D_{\text{pose}}(\mathbf{A}, \mathbf{G}) = \|\mathbf{A} - \mathbf{G}\|_F$ is the Frobenius norm between pose matrices $\mathbf{A}$ and $\mathbf{G}$. The objective of the following optimization is to find a setting of the kinematic parameters $\mathbf{R}$ that produces Cartesian poses consistent with $\mathbf{p}^*_{1:T}$.

$$\mathbf{R}^* = \underset{\mathbf{R}}{\operatorname{argmin}} \sum_{t=1}^{T} D_{\text{pose}}(K_{\mathbf{R}}(\mathbf{j}_t), \mathbf{p}^*_t) \tag{5}$$

Note, we are using $\mathbf{p}^*_{1:T}$, the output of the previous subroutine, as a target in this optimization. Since these were learned without considering kinematic structure, they may be inaccurate and therefore, the full model optimization (described in the next section), which simultaneously optimizes all model parameters, is needed to correct these errors.

C. Full Model Learning

Finally, we optimize all model parameters simultaneously. This optimization is very similar to that of the *Camera and Structure* learning (Section III-A). The error is also between the generated and observed pixel coordinates $\mathbf{I}_t$. However instead of using the truncated model (from $\mathbf{p}_t$ to $\mathbf{I}_t$) we consider the full model (from $\mathbf{j}_t$ to $\mathbf{I}_t$).

$$\mathbf{R}^f, \mathbf{F}^f, \mathbf{V}^f_{1:c} = \underset{\mathbf{R},\mathbf{F},\mathbf{V}_{1:c}}{\operatorname{argmin}} \sum_{t=1}^{T}\sum_{i=1}^{c} D_{\text{pixel}}(C_{\mathbf{V}_i}(S_{\mathbf{F}}(K_{\mathbf{R}}(\mathbf{j}_t))), \mathbf{I}_t^{(i)}) \tag{6}$$

The outputs of the previous two subroutines ($\mathbf{R}^*$, $\mathbf{F}^*$, $\mathbf{V}^*_{1:c}$) serve as the initialization for this optimization. We found that all 3 subroutines of this optimization are needed to ensure consistent convergence. Table II compares to DVS-OnlyFull, an ablation of our method that does not use the other 2 subroutines for learning, and we find that it sometimes fails to learn accurate models, especially as we reduce the amount of training data.

IV. Inference

After learning the model parameters, the model can predict the coordinates $\mathbf{I}_t$ of the image features given the robot's joint positions $\mathbf{j}_t$. However, in order to use this model for visual servoing, we need to be able to find the joint positions $\hat{\mathbf{j}}_t$ such that the image features will be located at some desired target locations $\hat{\mathbf{I}}_t$ in the image. We use an inference-via-optimization approach to infer $\hat{\mathbf{j}}_t$:

$$\hat{\mathbf{j}}_t = \underset{\mathbf{j}_t}{\operatorname{argmin}}\ D_{\text{pixel}}\big(f(\mathbf{j}_t),\ \hat{\mathbf{I}}_t) \tag{7}$$

where f is our learned model.

We can then use the above inference to control the robot. We first infer the current joint positions $\mathbf{j}_t$ from the current observations $\mathbf{I}_t$. Then, we select desired locations of the image features $\hat{\mathbf{I}}_t$ and infer the corresponding joint positions $\hat{\mathbf{j}}_t$. Finally, we command the robot to move by $\hat{\mathbf{j}}_t - \mathbf{j}_t$ to match those desired image feature coordinates. This optimization is implicitly solving optimizations corresponding to Inverse Kinematics (IK) [3] and Perspective-n-Point [10] problems.

V. Experiments

We conducted several experiments that demonstrate the accuracy, data-efficiency, speed, and flexibility of our approach. We first present a quantitative evaluation comparing our method, ablations of our method, and neural baselines. The results show that our approach learns very accurate models more data-efficiently than neural baselines. Then, we show the speed of our generative model, demonstrating that our approach is usable for real-time applications. Next, we show that our method can detect and quickly adapt to changes in the robot-camera setup via online learning. Finally, we evaluate our method's servoing accuracy, showing comparable accuracy to VISP [26], a standard calibrated method for visual servoing.

A. Data-Efficiency

We first demonstrate the data-efficiency of our method on real and simulated robot environments. Table II shows a quantitative evaluation of our method compared to neural network baselines and ablations of our method. Each method is trained on the indicated number of training samples (pairs of $(\mathbf{j}_t, \mathbf{I}_t)$) and tested on a held-out data set of 100 samples. We report the average error between the predicted and observed pixel coordinates (for visual features that were observed). We compare with 4 neural architecture baselines. **NN1**: Single hidden layer (200 units), Tanh activation. **NN2**: Single hidden layer (200 units), ReLU activation. **NN3**: Two hidden layers (100, 50 units), Tanh, ReLU activations. **NN4**: Two hidden layers (100, 50 units), ReLU, ReLU activations. We also compare with 2 ablations of our model. **DVS-NoFull**: *Camera and Feature Structure Learning* (Section III-A) and *Kinematic Parameter Learning* are used but not *Full Model Learning* (Section III-C). **DVS-OnlyFull**: Only *Full Model Learning* (Section III-C) is used. Our full architecture is denoted **DVS**, which uses all 3 subroutines for learning the model parameters. We find that our method consistently outperforms the baselines in all environments. The neural networks are unable to match the performance of our model given the same amount of data.

This is due in part to the concise parametrization of our model, which prevents overfitting the limited training data. The neural architecture lacks the structure of our model and is significantly overparametrized. Note that for an n degree of freedom (DOF) robot with m tracked features that are observed in c cameras, our model's parameters are: 6 for the robot base frame, $4n$ for DH parameters for all joints, $3m$ for 3D coordinates of each of the features, $6c$ for the camera extrinsics and $4c$ for camera intrinsics, totaling $6+4n+3m+10c$ parameters. In our experiments where we have a 6DoF robot, 12 features, and 2 cameras, the total number of parameters is 86. In comparison, a neural architecture with just 50 hidden units already has 2798 parameters.

We also compare with two ablations of our method, DVS-NoFull and DVS-OnlyFull, which don't use the full learning algorithm but only some subset of the 3 subroutines. DVS-OnlyFull, the ablation that only uses the *Full Model Learning* subroutine (Section III-C), can sometimes match the performance of our full method DVS when 75 or 100 training samples are available. However, when only 50 samples are available the full method outperforms the ablations, illustrating the need for all 3 subroutines.

B. Speed

For our architecture to be practically useful, it must be fast enough to be used for real-time servoing. In Table III, we provide times for the forward, gradient, and inference computations in our model. Real-time servoing systems, such as VISP [26], use gradient-based servoing. Computing a gradient through our full model takes 16.1ms implying that our model supports 60+ FPS servoing, which would not be the bottleneck in most visual servoing systems given standard frame rates of cameras and frame rates of object/feature detectors that are often used for vision-based control.

C. Online Learning

Most robotic systems assume that kinematic and camera calibrations will be performed once during setup and will remain fixed over the course of the robot's use. These system often do not monitor these calibrations, so if they were to change, this may only be detected in downstream task performance. For example, imagine a robot that uses an overhead camera to detect objects and plan grasps. If the camera's extrinsics were to drift by 1cm, the resulting grasps would be offset by 1cm, which would degrade the overall grasping performance. We demonstrate that by using our model, we can detect these changes to the system and adapt to them via online learning. We evaluate our method's ability to adapt to these changes by first learning the model parameters with an initial configuration of the robot-camera setup, then modifying the setup and measuring the model's reconstruction error as it sequentially receives new observations and updates the model parameters online. The reconstruction error is computed on a held-out data set collected in the new setup.

First, we consider modifications to the cameras (adding a new camera and moving an existing camera), which might happen accidentally if a camera is bumped or intentionally if another viewpoint is needed for a certain task. (We conducted this experiment on the UR5 real environment). Figure 3(a) shows the reconstruction error as a function of the number of new samples available for the case where a third camera is added and the case where one of existing two cameras is moved. With just 2 new samples, the model is sufficiently accurate to begin servoing again and continues to improve its accuracy as more samples become available.

Next, we consider modifications to the structure of the visual features. Consider the setting where a robot grasps a peg and wants to insert it into a hole. To do this accurately, we would track and servo using visual features on the peg to align it to the hole. This would require learning the 3D structure of the new visual features on the peg. We evaluate our method's ability to learn new visual features, by attaching a novel marker to the robot's end-effector. (We conducted this experiment on the UR5 real environment). Figure 3(b) shows the reconstruction error (of the new visual features) as

TABLE II
RECONSTRUCTION ERROR OF LEARNED MODELS ON EACH OF THE 4 ENVIRONMENTS, VARYING THE TRAINING DATA SET SIZE

		Reconstruction Error (Test)						
		Neural Baselines				DurableVS (DVS)		
Environment	**# Samples**	NN1	NN2	NN3	NN4	DVS-NoFull	DVS-OnlyFull	DVS
UR5 (sim)	50	26.45	13.87	70.51	12.43	10.91	8.92	**0.64**
	75	15.00	14.35	85.45	9.15	11.40	4.36	**0.64**
	100	20.45	11.35	79.34	10.36	12.54	3.04	**0.63**
Xarm (sim)	50	39.27	27.15	27.82	23.00	7.54	31.02	**0.37**
	75	38.24	19.47	29.05	16.08	7.58	0.36	**0.36**
	100	33.80	14.76	16.90	14.01	9.40	**0.36**	**0.36**
UR5 (real)	50	60.28	3.56	29.61	4.48	6.46	4.64	**0.36**
	75	60.27	23.00	28.99	3.96	9.28	4.25	**0.37**
	100	60.27	3.62	59.69	5.58	6.53	4.84	**0.36**
Baxter (real)	50	16.07	13.54	16.25	14.76	7.87	12.17	**2.40**
	75	12.61	11.02	14.48	8.88	12.36	2.21	**2.20**
	100	13.83	9.57	11.31	7.93	7.58	**2.01**	**2.00**

TABLE III
SPEED OF QUERIES IN DURABLEVS

Query	**Time (s)**	**FPS**
Forward Kinematics	0.0012	833
Forward Camera/Feature	0.0014	714
Forward Full Model	0.0025	400
Gradient Full Model	0.0161	62
Gradient Kinematics	0.0126	79
Gradient Camera/Feature	0.0091	110
Infer Pose from Image	0.0005	2000
Infer Joints from Pose	0.1012	10
Infer Joints from Image	0.1025	10

a function of the number of new samples available. With 2 samples, the model is already very accurate. One explanation for this efficiency is to think that if we had two cameras properly calibrated, the new features could be triangulated to 3D, given pixel coordinates. With more views, the error in the triangulation reduces. Note, this triangulation is not done explicitly, only implicitly through the optimization.

Finally, we may want to relearn the robot's kinematic model, for example, if a joint is damaged or if we modify the end-effector tool e.g. swap the gripper. (Since we could not modify the real robot kinematics, we conducted this experiment on the simulated UR5 robot by adding random noise to the robot's link lengths.) In Figure 3(c), we show the reconstruction error as a function of the number of new samples available. Compared to re-learning camera and feature structure parameters, re-learning the kinematic parameters require more new samples. However, within approximately 25 samples the model has adapted and is very accurate again.

These results demonstrate that when there are unexpected changes to the robot-camera setup (e.g. bumped cameras, new visual features, damaged robot), modules of the generative model can be quickly relearned to adapt to those changes. But how do we detect these unexpected changes? We can detect these changes by comparing the expected image feature coordinates (as predicted by the learned generative model) with the actual observations. During normal operation, these should match very closely, however when there are unexpected changes to the system there will be discrepancies between the expectation and observation. We use this as a signal that model parameters need to be updated.

D. Servoing Accuracy

Finally, we demonstrate that our architecture enables accurate visual servoing. Figure 3d show a comparison between our method and VISP [26], a popular library implementing visual servoing. In this evaluation, we first randomly select target locations for the visual features and then allow each method to servo to the target. We measure the error between the target and actual pixel coordinates at each time step. (We average over 50 runs with different randomly selected targets). We found that both DurableVS and VISP achieve sub-pixel accuracy. Our method slightly outperforms VISP, likely due to inaccuracies in camera calibration, which further motivates the value of learning as opposed to assuming a fixed camera calibration[1]. In these experiments, it appears DurableVS converges to the targets more rapidly than VISP. However, we do not claim that DurableVS is faster than VISP in general, as VISP's convergence rate depends on many factors. We provide this data to show that DurableVS can be competitive with VISP (out of the box) on a real robot, in addition to requiring no calibration and automatically recovering from a broad class of calibration errors.

VI. RELATED WORK

A. Kinematic & Camera Calibration

Robotic systems often assume the robot's kinematic model and any cameras it uses are calibrated. Kinematic calibration often needs precisely manufactured devices or manual alignment procedures [16], [18], [33] which can be time-consuming. Camera calibration needs a pattern to be placed at different locations in view of the camera [4], [32], [38], [42] and this process must be repeated whenever the camera is

[1]We used a standard camera calibration routine provided in ROS [32] and validated its accuracy

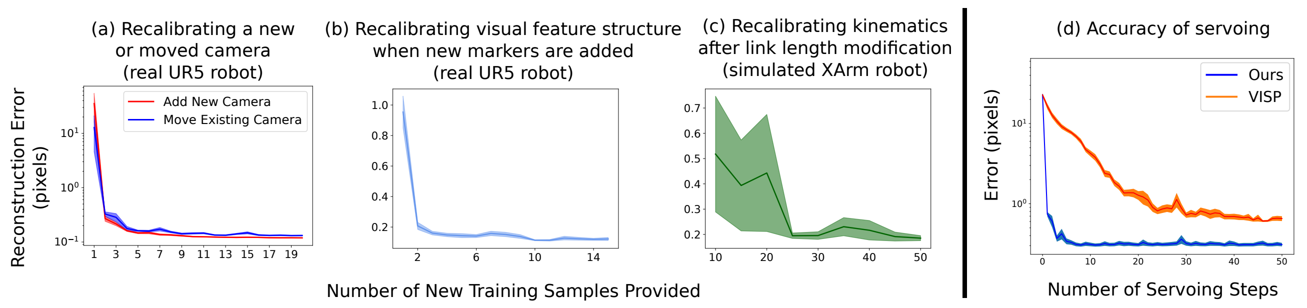


Fig. 3. (a,b,c) We evaluate our method's ability to adapt to changes to the robot-camera setup by measuring the model's reconstruction error as it sequentially receives new observations and updates the model parameters online. (a) On the real UR5, we measure the reconstruction error in both the case where we add a new camera and the case where we move an existing camera. (b) On the real UR5, we measure the reconstruction error when we attach new visual features to the end-effector. (c) On a simulated UR5, we measure the reconstruction error when we modify the link lengths of the robot, simulating a change to the robot kinematics. (d) We compare the accuracy of our method with VISP [26] on servoing to randomly select target coordinates of the visual features.

moved. Other prior works propose methods for learning kinematics [2], [7], [18], [33], [34] or self-calibration of cameras [9], [11], [15], [22], [39]. However, none of these methods can learn both kinematic and camera models, unsupervised.

Independent kinematic and camera calibration can also negatively affect accuracy of visual servoing since errors in the kinematic calibration can lead to errors in camera calibration. Many prior works explore the benefits of simultaneous kinematics and camera calibration [21], [39], [43] in avoiding these compounded errors. However, these calibration approaches are aimed at identifying the veridical system parameters and thus cannot be learned without supervision.

Finally, kinematics and camera calibration must be repeated when even small changes are made to the robot-camera system. As demonstrated in our experiments, our approach can quickly adapt to such changes online, enabling the system to continue to function without interruption or intervention.

B. Visual Servoing

Visual servoing refers to techniques that use visual feedback to guide control. Commonly used methods for visual servoing are image-based visual servoing (IBVS) and position-based visual servoing (PBVS). In general, these methods seek to move certain tracked image features to desired target locations. Most proposed visual-servoing approaches depend on calibration of the robot's kinematic model and cameras [5], [8], [28], [28], [36]. To relax this assumption some methods directly estimate the image-joint Jacobian, capturing how changes to the robot's joint positions lead to changes in image features [17], [19], [27], [30], [41]. Unlike differential approaches, our method learns a structured model of the system.

Visual Servoing Platform (VISP) [26] is a popular library implementing various visual servoing control laws. Part of our contribution in this work is the Durable Visual Servoing (DVS) Python [40] package which implements our architecture in PyTorch [29]. This package includes a demo of data-collection, learning, and visual servoing with the learned model in a simulated PyBullet [6] environment. In addition, we provide tutorials to enable researchers to apply our system to their applications and to build upon our architecture.

C. Learning Vision-based Control

Recent work on learning control of robots using visual feedback has focused on deep learning approaches. [35] learns viewpoint-invariant visual servoing but requires pre-training in simulated environments before transferring to real environments. [23] trains perception and control systems simultaneously using an end-to-end neural network and demonstrates that this can be applied to perform a wide range of tasks. However, as is often the case with neural networks, they demand a large amount of training data and time. With our method, models can be learned from significantly fewer samples and less time than these neural methods.

In future work, we plan to explore the combination of our structured generative model with deep learning approaches. We believe that this combination can improve the overall data-efficiency of end-to-end learning of robotic tasks like grasping [24], [31], drone flying [12], and object manipulation/interaction [1]. Consider if the deep learned system could robustly detect visual features and plan the path of those image features such that the robot performs the desired task. Our architecture could then take this as input and infer the necessary joint commands to perform the task. In this paper, we used Aruco markers [13] attached to the end-effector as the visual features. By combining our architecture with deep learning, we could develop an overall more general system.

VII. Conclusion

In this paper, we presented an approach to learning visual servoing based on structured generative models. We have shown that given an arbitrary uncalibrated robot observed through an arbitrary number of uncalibrated cameras, our system can learn accurate visual servoing. This is done completely unsupervised and extremely data-efficiently (in less than a minute). Learning can continue online, enabling the system to detect and recover from changes to the robot-camera system. The problem we have addressed in this work is broadly applicable to many vision-based robotics domains and tasks. We hope that our contribution can enable more data-efficient and robust systems that can operate in real-world settings with no human supervision.

References

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in neural information processing systems*, pages 5074–5082, 2016.

[2] Christopher G Atkeson. Learning arm kinematics and dynamics. *Annual review of neuroscience*, 12(1):157–183, 1989.

[3] Patrick Beeson and Barrett Ames. Trac-ik: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 928–935. IEEE, 2015.

[4] Gary Bradski and Adrian Kaehler. Opencv. *Dr. Dobb's journal of software tools*, 3, 2000.

[5] François Chaumette, Seth Hutchinson, and Peter Corke. Visual servoing. In *Springer Handbook of Robotics*, pages 841–866. Springer, 2016.

[6] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation in robotics, games and machine learning, 2017.

[7] Aaron D'Souza, Sethu Vijayakumar, and Stefan Schaal. Learning inverse kinematics. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 298–303. IEEE, 2001.

[8] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *ieee Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[9] Olivier D Faugeras, Q-T Luong, and Stephen J Maybank. Camera self-calibration: Theory and experiments. In *European conference on computer vision*, pages 321–334. Springer, 1992.

[10] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] Clive S Fraser. Digital camera self-calibration. *ISPRS Journal of Photogrammetry and Remote sensing*, 52(4):149–159, 1997.

[12] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955. IEEE, 2017.

[13] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[14] Richard Hartenberg and Jacques Danavit. *Kinematic synthesis of linkages*. New York: McGraw-Hill, 1964.

[15] Elsayed Hemayed. A survey of camera self-calibration. In *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.*, pages 351–357. IEEE, 2003.

[16] John M Hollerbach and Charles W Wampler. The calibration index and taxonomy for robot kinematic calibration methods. *The international journal of robotics research*, 15(6):573–591, 1996.

[17] Koh Hosoda, Katsuji Igarashi, and Minoru Asada. Adaptive hybrid control for visual and force servoing in an unknown environment. *IEEE Robotics & Automation Magazine*, 5(4):39–43, 1998.

[18] Milan Ikits and John M Hollerbach. Kinematic calibration using a plane constraint. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3191–3196. IEEE, 1997.

[19] Martin Jagersand, Olac Fuentes, and Randal Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 2874–2880. IEEE, 1997.

[20] Danica Kragic and Henrik I Christensen. Survey on visual servoing for manipulation. In *USENIX Technical Conference*, 2002.

[21] Rainer Kümmerle, Giorgio Grisetti, and Wolfram Burgard. Simultaneous parameter calibration, localization, and mapping. *Advanced Robotics*, 26(17):2021–2041, 2012.

[22] Reimar K Lenz and Roger Y Tsai. Calibrating a cartesian robot with eye-on-hand configuration independent of eye-to-hand relationship. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):916–928, 1989.

[23] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[24] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[25] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[26] Éric Marchand, Fabien Spindler, and François Chaumette. Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics & Automation Magazine*, 12(4):40–52, 2005.

[27] Amir massoud Farahmand, Azad Shademan, and Martin Jagersand. Global visual-motor estimation for uncalibrated visual servoing. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1969–1974. IEEE, 2007.

[28] Kartik Mohta, Vijay Kumar, and Kostas Daniilidis. Vision-based control of a quadrotor for perching on lines. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3130–3136. IEEE, 2014.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[30] Jenelle Armstrong Piepmeier, Gary V McMurray, and Harvey Lipkin. Uncalibrated dynamic visual servoing. *IEEE Transactions on Robotics and Automation*, 20(1):143–147, 2004.

[31] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[32] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[33] Jean-Michel Renders, Eric Rossignol, Marc Becquet, Raymond Hanus, et al. Kinematic calibration and geometrical parameter identification for robots. *IEEE Transactions on robotics and automation*, 7(6):721–732, 1991.

[34] Matthias Rolf, Jochen J Steil, and Michael Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3):216–229, 2010.

[35] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. Sim2real viewpoint invariant visual servoing by recurrent control. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4691–4699, 2018.

[36] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. springer, 2016.

[37] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[38] Roger Y Tsai and Reimar Lenz. Review of the two-stage camera calibration technique plus some new implementation tips and some new techniques for center and scale calibration. In *Optical Society of America, Topical Meeting on Machine Vision*, 1980.

[39] Roger Y Tsai and Reimar K Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Transactions on robotics and automation*, 5(3):345–358, 1989.

[40] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[41] Billibon H Yoshimi and Peter K Allen. Active, uncalibrated visual servoing. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 156–161. IEEE, 1994.

[42] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[43] Hanqi Zhuang, Kuanchih Wang, and Zvi S Roth. Simultaneous calibration of a robot and a hand-mounted camera. *IEEE Transactions on Robotics and Automation*, 11(5):649–660, 1995.