Learning Latent Space Dynamics for Tactile Servoing

Giovanni Sutanto[1,2,3], Nathan Ratliff[1], Balakumar Sundaralingam[1,4], Yevgen Chebotar[1,3], Zhe Su[2,3], Ankur Handa[1] and Dieter Fox[1,5]

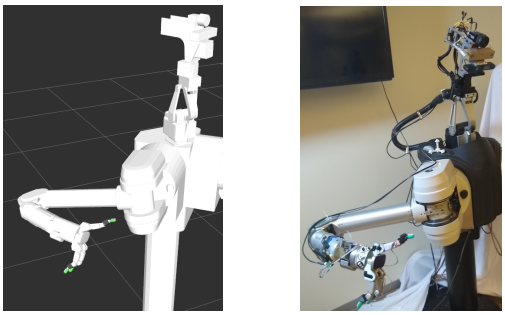
***Abstract*— To achieve a dexterous robotic manipulation, we need to endow our robot with tactile feedback capability, *i.e.* the ability to drive action based on tactile sensing. In this paper, we specifically address the challenge of tactile servoing, *i.e.* given the current tactile sensing and a target/goal tactile sensing — memorized from a successful task execution in the past — what is the action that will bring the current tactile sensing to move closer towards the target tactile sensing at the next time step. We develop a data-driven approach to acquire a dynamics model for tactile servoing by learning from demonstration. Moreover, our method represents the tactile sensing information as to lie on a surface — or a 2D manifold — and perform a manifold learning, making it applicable to any tactile skin geometry. We evaluate our method on a contact point tracking task using a robot equipped with a tactile finger.**

I. Introduction

The ability to adapt actions based on tactile sensing is the key to robustly interact and manipulate with objects in the environment. Previous experiments have shown that when the tactile-driven control is impaired, humans have difficulties performing even basic manipulation tasks [1], [2]. Hence, we believe that equipping robots with tactile feedback capability is important to make progress in robotic manipulation.

In line with this direction, recently a variety of tactile sensors [3], [4], [5], [6] have been developed and used in robotics research community, and researchers have designed several tactile-driven control — or popularly termed as *tactile servoing* — algorithms. However, many tactile servoing algorithms were designed for specific kinds of tactile sensor geometry, such as a planar surface [7] or a spherical surface [8], therefore they do not apply to the broad class of tactile sensors in general. For example, if we would like to equip a robot with a tactile skin of arbitrary geometry or if there is a change in the sensor geometry due to wear or damage, we will need a more general tactile servoing algorithm.

In this paper, we present our work on a learning-based tactile servoing algorithm that does not assume a specific sensor geometry. Our method comprises three steps. At the core of our approach, we treat the tactile skin as a manifold, hence first we perform an offline neural-network based manifold learning, to learn a latent space representation which encodes the essence of the tactile sensing information. Second, we learn a latent space dynamics model from human demonstrations. Finally, we deploy our model to perform an online control — based on both the current and target tactile signals — for tactile servoing on a robot.



(a) Simulated Robot (b) Real Robot

Fig. 1. Learning tactile servoing platform.

Our contribution is twofold: First, we utilize manifold learning to impose an Euclidean structure in the latent space representation of tactile sensing, such that the control in this latent space becomes straightforward. Second, we train a single model that is able to do both forward dynamics and inverse dynamics prediction using the same demonstration dataset, which is more data-efficient than training separate models for the forward and inverse dynamics.

This paper is organized as follows. Section II provides some related work. Section III presents the model that we use for learning tactile servoing from demonstration. We then present our experimental setup and evaluations in Section IV. Finally, we discuss our results and future work in Section V.

II. Related Work

Our work is mostly inspired by previous works on learning control and dynamics in the latent space [9], [10]. Both of these works learn a latent space representation of the state, and also learn a dynamics model in the latent space. Watter et al. [10] designed the latent space's state transition model to be locally linear, such that a stochastic optimal control algorithm can be directly applied to the learned model for control afterwards. Byravan et al. [9] designed the latent space to represent SE(3) poses of the tracked objects in the scene, and the transition model is simply the SE(3) transformations of these poses. Control in [9] is done by gradient-following of the Euclidean distance between the target and current latent space poses with respect to action.

In this work, we train a latent space dynamics model that takes latent space representation of the current tactile sensing and applied action, and predicts the latent space representation of the next tactile sensing, which is termed as *forward dynamics*. Since we use the model for control, *i.e.* tactile servoing, it is also essential that we can compute actions, given both the current and target tactile sensing — termed as *inverse dynamics*. Previous work [11] learns

[1]NVIDIA, USA.
[2]Autonomous Motion Department, MPI-IS, Tübingen, Germany.
[3]University of Southern California, Los Angeles, USA.
[4]University of Utah, Salt Lake City, USA.
[5]University of Washington, Seattle, WA, USA.

This research was supported in part by NVIDIA Research, National Science Foundation grants IIS-1205249, IIS-1017134, EECS-0926052, the Office of Naval Research, the Okawa Foundation, and the Max-Planck-Society.

separate models, one for the forward dynamics, and one for the inverse dynamics model, for a poking task. In contrast to this, in our work we train a single model for both forward and inverse dynamics.

In terms of latent space representation, our work is inspired by the work of Hadsell *et al.* [12], where they use a Siamese neural network and construct a loss function such that similar data points are close to each other in the latent space and dissimilar data points are further away from each other in the latent space. In this work, we also use a Siamese neural network, however we employ a loss function that performs Multi-Dimensional Scaling (MDS) [13], such that the first two dimensions of the latent space represent the 2D map of the contact point on the tactile skin surface. Our third dimension in the latent space represents the degree of contact applied on the skin surface, *i.e.* how much pressure was applied at the point of contact.

Regarding tactile servoing, besides the previous works [7], [8] which have been mentioned in Section I, Su *et al.* [14] designed a heuristic for tactile servoing with a tactile finger [3]. Our work treats the tactile sensor as a general manifold, hence the method shall apply to any tactile sensors.

Previously, learning tactile feedback has been done through reinforcement learning [15] or a combination of imitation learning and reinforcement learning [16], [17]. Sutanto et al. [18] learns a tactile feedback model for a trajectory-centric reactive policy. In this work, we learn a tactile servoing policy indirectly by learning a latent space dynamics model from demonstrations. As we engineer the latent space to be Euclidean by performing MDS and maintaining contact degree information, the inverse dynamics control action can be computed analytically, given both the current and target latent states. Hence, our method does not require reinforcement learning to learn the desired behavior.

III. Data-Driven Tactile Servoing Model

A. Tactile Servoing Problem Formulation

Given the current tactile sensing $\mathbf{s}_t$ and the target tactile sensing $\mathbf{s}_T$, the objective is to find the action $\mathbf{a}_t$ which will bring the next tactile sensing $\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t)$ closer to $\mathbf{s}_T$, which in the optimal case can be written as:

$$\mathbf{a}_t^* = \arg\min_{\mathbf{a}_t} d(\mathbf{f}(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_T) \tag{1}$$

B. Latent Space Representation

If the distance metric d is a squared $\mathcal{L}_2^2$ distance of two states, which lie on a Euclidean space and if $\mathbf{f}$ is smooth, then inverse dynamics $\mathbf{a}_t$ can be computed as proportional to $-\frac{\partial d}{\partial \mathbf{a}_t}$. Moreover, for some $\mathbf{f}$, the $\mathbf{a}_t^*$ in Eq. 1 can be computed analytically, at the condition $\frac{\partial d}{\partial \mathbf{a}_t} = \mathbf{0}$. Unfortunately, both $\mathbf{s}_{t+1}$ and $\mathbf{s}_T$ may not lie on a Euclidean space.

On the other hand, there seems to be some natural characterization of tactile sensing, such as the contact point and the degree of contact pressure applied at the point. The contact point in particular is a 3D coordinate which lies on the skin surface. Obviously we know that the skin surface is not Euclidean, *i.e.* we cannot go from the current contact point to the target contact point by simply following the vector between them because then it may be off the skin surface while doing so[1]. However, if we are able to flatten the skin surface in 3D space into a 2D surface, then traversing between the two contact points translates into following the vector from one 2D point to the other on the 2D surface, which ensures that the intermediate points being traversed are all still on the 2D surface. Fortunately, there has been a method of mapping/embedding from a 3D surface into a 2D surface, called Multi-Dimensional Scaling (MDS) [13].

In this paper, we choose the latent space embedding to be three-dimensional[2]:

1) The first two dimensions of the latent space –called the x and y dimensions of the latent space– corresponds to the 2D embedding of the 3D contact point on the tactile skin surface.
2) The third dimension (the z dimension) of the latent space represents the degree of contact pressure applied at the contact point.

We understand that the above representation can only represent a contact as a single 3D coordinate in the latent space. Therefore, it will not be able to capture the richer set of features, such as an object's edges and orientations, etc. Tactile servoing for edge tracking is left for a future work.

C. Embedding Function

We call the latent state representation of a tactile sensing $\mathbf{s}$ as $\mathbf{z}$, and we define the distance metric d as a squared $\mathcal{L}_2^2$ distance in the latent space between the embeddings of $\mathbf{s}_{t+1}$ and $\mathbf{s}_T$ by the embedding function $\mathbf{z} = \mathbf{f}_{enc}(\mathbf{s})$, as follows[3]:

$$d(\mathbf{s}_{t+1}, \mathbf{s}_T) = \|\mathbf{z}_{t+1} - \mathbf{z}_T\|^2 \tag{2}$$

We represent the embedding function $\mathbf{f}_{enc}$ by the encoder part of an auto-encoder neural network.

For achieving the latent space representation as mentioned in III-B, we impose the following structure:

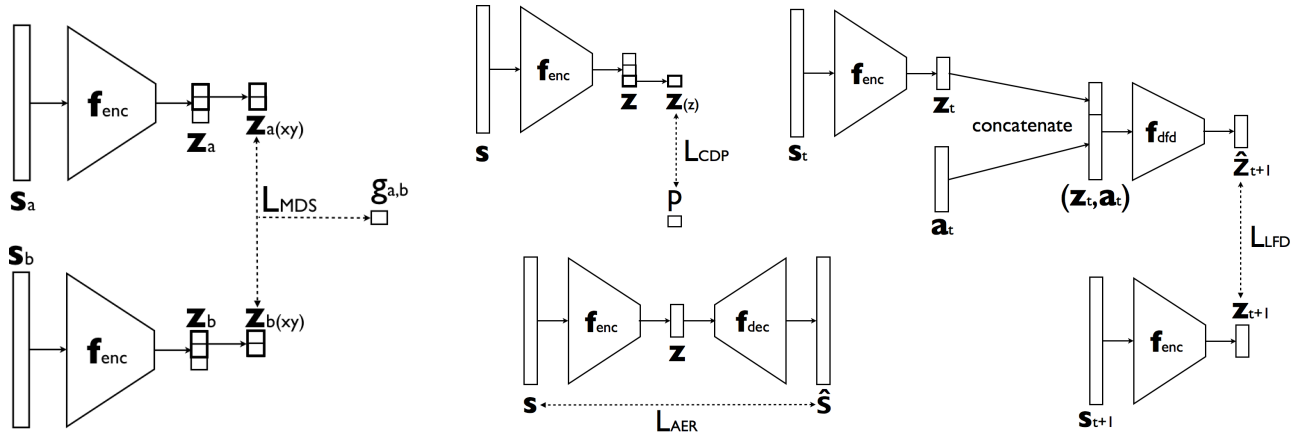
1) We would like to map points on a surface in 3D space into 2D coordinates. Essentially this can be described as a 2D manifold embedded in 3D space. For such a manifold, the notion of distance between any pair of two 3D points on the manifold is given by the geodesics, i.e. the curve of the shortest path on the surface. For this mapping, we would like to preserve these pairwise geodesic distances in the resulting 2D map. That is, for pairs of data points $\{\{\mathbf{s}_a^{(1)}, \mathbf{s}_b^{(1)}\}, \{\mathbf{s}_a^{(2)}, \mathbf{s}_b^{(2)}\}, \ldots, \{\mathbf{s}_a^{(K)}, \mathbf{s}_b^{(K)}\}\}$, we want to acquire a latent space embedding via the embedding function $\mathbf{z} = \mathbf{f}_{enc}(\mathbf{s})$ to get the latent space pairs $\{\{\mathbf{z}_a^{(1)}, \mathbf{z}_b^{(1)}\}, \{\mathbf{z}_a^{(2)}, \mathbf{z}_b^{(2)}\}, \ldots, \{\mathbf{z}_a^{(K)}, \mathbf{z}_b^{(K)}\}\}$ whose distance in the x and y dimensions is as close as possible to the pairwise geodesic distance. Therefore we define the loss function [19]:

$$L_{MDS} = \sum_{k=1}^{K} \left\| \left\| \mathbf{z}_{a(xy)}^{(k)} - \mathbf{z}_{b(xy)}^{(k)} \right\| - g_{a,b}^{(k)} \right\|^2 \tag{3}$$

[1]The correct way of traversing from a contact point to the other is by following the geodesics between the two points on the skin surface.

[2]Here we assume that there exists a mapping from a tactile sensing $\mathbf{s}$ into the 3D contact point on the tactile skin surface as well as a mapping from $\mathbf{s}$ into the degree of contact pressure information.

[3]Subscripts in Eq. 2 corresponds to time indices.



(a) L_{MDS} (b) L_{AER}, L_{CDP}, and L_{LFD}

Fig. 2. Neural Network diagrams and its loss functions (drawn as dotted lines). Inverse dynamics loss function L_{ID} is not illustrated here.

K is the number of data point pairs which is quadratic in the total number of data points N. $g_{a,b}^{(k)}$ is the geodesic distance between the two data points in the k-th pair. The pairwise geodesic distance between any two data points is approximated using the shortest path algorithm on a sparse distance matrix of M-nearest-neighbors of each data point. We use M-nearest-neighbors because the space is not 2D-Euclidean *globally* due to skin curvature, but it is *locally* 2D-Euclidean –i.e. flat– on a small neighborhood (a small patch) on the skin. The computation result is stored as a symmetric dense approximate geodesic distance matrix of size $N \times N$ before the training begins. The pairwise loss function in Eq. 3 is applied by using a Siamese neural network as depicted in Figure 2(a).

2) Encoding of the z dimension of the latent space with the contact pressure information p. This is done by imposing the following loss function:

$$L_{CDP} = \sum_{n=1}^{N} \left\| p^{(n)} - \mathbf{z}_{(z)}^{(n)} \right\|^2 \tag{4}$$

While we have the ground truth for the z dimension of the latent state, i.e. p, we do not have the ground truth for the x and y dimensions. We have the 3D coordinate of the data point on the skin[4] — which is used to compute the sparse distance matrix of M-nearest-neighbors of each data point — but we do not know how it is mapped to the x and y dimensions of the latent space, and this is our reason for using an auto-encoder neural network representation. The auto-encoder reconstruction loss is:

$$L_{AER} = \sum_{n=1}^{N} \left\| \mathbf{f}_{dec}(\mathbf{f}_{enc}(\mathbf{s}^{(n)})) - \mathbf{s}^{(n)} \right\|^2 \tag{5}$$

with $\mathbf{f}_{enc}$ is the encoder/embedding function, and $\mathbf{f}_{dec}$ is the decoder/inverse-embedding function.

D. Latent Space Forward Dynamics (LFD)

We assume the latent space forward dynamics as follows:

$$\dot{\mathbf{z}}_t = \mathbf{f}_{fd}(\mathbf{z}_t, \mathbf{a}_t) \tag{6}$$

[4]For BioTacs, these 3D coordinates can be computed from electrode values, by using the point of contact estimation model presented in [20].

and numerical integration gives us the discretized version:

$$\mathbf{z}_{t+1} = \mathbf{z}_t + \dot{\mathbf{z}}_t \Delta t = \mathbf{z}_t + \mathbf{f}_{fd}(\mathbf{z}_t, \mathbf{a}_t)\Delta t = \mathbf{f}_{dfd}(\mathbf{z}_t, \mathbf{a}_t) \tag{7}$$

There are two possibilities of $\mathbf{f}_{fd}$ as follows:

1) Locally Linear (LL) LFD Model:

$$\dot{\mathbf{z}}_t = \mathbf{f}_{fd}(\mathbf{z}_t, \mathbf{a}_t) = \mathbf{A}_t \mathbf{z}_t + \mathbf{B}_t \mathbf{a}_t + \mathbf{c}_t \tag{8}$$

where $\mathbf{A}_t$, $\mathbf{B}_t$, and $\mathbf{c}_t$ is predicted by a fully-connected neural network (fcnn) from input $\mathbf{z}_t$, as follows:

$$\begin{bmatrix} \text{vec}(\mathbf{A}_t) \\ \text{vec}(\mathbf{B}_t) \\ \mathbf{c}_t \end{bmatrix} = \mathbf{h}_{fcnnLL}(\mathbf{z}_t) \tag{9}$$

with $\text{vec}(\mathbf{A}_t)$ and $\text{vec}(\mathbf{B}_t)$ are the vectorized representation of $\mathbf{A}_t$ and $\mathbf{B}_t$, respectively [10].

2) Non-Linear (NL) LFD Model:

$$\dot{\mathbf{z}}_t = \mathbf{f}_{fd}(\mathbf{z}_t, \mathbf{a}_t) = \mathbf{h}_{fcnnNL}(\begin{bmatrix} \mathbf{z}_t \\ \mathbf{a}_t \end{bmatrix}) \tag{10}$$

We would like to be able to predict the forward dynamics in the latent space, so we use the following loss function:

$$L_{LFD} = \sum_{t=1}^{H} \left\| \mathbf{z}_{t+1} - \mathbf{f}_{enc}(\mathbf{s}_{t+1}) \right\|^2 \tag{11}$$

with $\mathbf{z}_{t+1}$ is computed from Eq. 6 and 7. For additional robustness, we can also do chained predictions for C time steps ahead and sum up the loss function in Eq. 11 for these chains, similar to the work by Nagabandi et al. [21].

E. Inverse Dynamics (ID)

Beside forward dynamics, we also found that the ability of the model to predict inverse dynamics to be essential for the purpose of action selection or control.

There are three possibilities of inverse dynamics model:

1) Locally Linear (LL) ID: Based on the locally linear LFD model [10] in section III-D.1, Eq. 6, 7, 8, we can setup a constrained optimal control problem:

$$\begin{aligned} \min_{\mathbf{a}_t, \mathbf{z}_{t+1}} \quad & \frac{1}{2} \left\| \mathbf{z}_T - \mathbf{z}_{t+1} \right\|^2 + \frac{\beta}{2} \left\| \mathbf{a}_t \right\|^2 \\ \text{s.t.} \quad & \mathbf{z}_{t+1} = \mathbf{z}_t + (\mathbf{A}_t \mathbf{z}_t + \mathbf{B}_t \mathbf{a}_t + \mathbf{c}_t)\Delta t \end{aligned} \tag{12}$$

whose solution is:

$$\mathbf{a}_{t,ID} = \mathbf{B}_t^{\mathrm{T}} \left(\mathbf{B}_t \mathbf{B}_t^{\mathrm{T}} + \frac{\beta}{\Delta t^2} \mathbf{I} \right)^{-1} \left(\frac{\mathbf{z}_T - \mathbf{z}_t}{\Delta t} - \mathbf{A}_t \mathbf{z}_t - \mathbf{c}_t \right) \tag{13}$$

2) Negative-Gradient (NG) ID: Based on the non-linear LFD model in section III-D.2, we can compute a gradient-based controller which minimizes the distance function $d = \|\mathbf{z}_t + \mathbf{f}_{fd}(\mathbf{z}_t, \mathbf{a}_t)\Delta t - \mathbf{z}_T\|^2$, that is [9]:

$$\mathbf{a}_{t,ID} = -\alpha \frac{\partial d}{\partial \mathbf{a}_t}\bigg|_{\mathbf{a}_t=\mathbf{0}} \tag{14}$$

with α is a positive contant that scales the gradient w.r.t. maximum allowed magnitude of $\mathbf{a}_t$

3) Neural Network Jacobian (NJ) ID: Based on the non-linear LFD model in section III-D.2, we can derive the following from Eq. 10 (dropping time index t for a moment):

$$\ddot{\mathbf{z}} = \begin{bmatrix}\mathbf{J}_z & \mathbf{J}_a\end{bmatrix}\begin{bmatrix}\dot{\mathbf{z}} \\ \dot{\mathbf{a}}\end{bmatrix} = \mathbf{J}_z\dot{\mathbf{z}} + \mathbf{J}_a\dot{\mathbf{a}} \tag{15}$$

which can be discretized into:

$$\begin{aligned}\mathbf{z}_{t+1} =& \mathbf{z}_t + \left(\frac{1}{\Delta t}\mathbf{I} + \mathbf{J}_{z_{t-1}}\right)\Delta t\,(\mathbf{z}_t - \mathbf{z}_{t-1}) \\ &+ \mathbf{J}_{a_{t-1}}\Delta t\,(\mathbf{a}_t - \mathbf{a}_{t-1})\end{aligned} \tag{16}$$

with $\mathbf{J}_{z_{t-1}}$ and $\mathbf{J}_{a_{t-1}}$ are the Jacobians of $\mathbf{h}_{fcnnNL}$[5] w.r.t. previous latent state $\mathbf{z}_{t-1}$ and previous action $\mathbf{a}_{t-1}$, respectively. Let us define $\bar{\mathbf{A}}_{t-1} = \left(\frac{1}{\Delta t}\mathbf{I} + \mathbf{J}_{z_{t-1}}\right)$, $\bar{\mathbf{B}}_{t-1} = \mathbf{J}_{a_{t-1}}$, and $\bar{\mathbf{c}}_{t-1} = -\bar{\mathbf{A}}_{t-1}\mathbf{z}_{t-1} - \bar{\mathbf{B}}_{t-1}\mathbf{a}_{t-1}$, Eq. 16 can be written as:

$$\mathbf{z}_{t+1} = \mathbf{z}_t + (\bar{\mathbf{A}}_{t-1}\mathbf{z}_t + \bar{\mathbf{B}}_{t-1}\mathbf{a}_t + \bar{\mathbf{c}}_{t-1})\Delta t \tag{17}$$

We can setup a constrained optimal control problem:

$$\begin{aligned}\min_{\mathbf{a}_t,\,\mathbf{z}_{t+1}} \quad & \frac{1}{2}\|\mathbf{z}_T - \mathbf{z}_{t+1}\|^2 + \frac{\beta}{2}\|\mathbf{a}_t\|^2 \\ \text{s.t.} \quad & \\ & \mathbf{z}_{t+1} = \mathbf{z}_t + (\bar{\mathbf{A}}_{t-1}\mathbf{z}_t + \bar{\mathbf{B}}_{t-1}\mathbf{a}_t + \bar{\mathbf{c}}_{t-1})\Delta t\end{aligned} \tag{18}$$

whose solution is:

$$\begin{aligned}\mathbf{a}_{t,ID} = \bar{\mathbf{B}}_{t-1}^{\mathrm{T}}\left(\bar{\mathbf{B}}_{t-1}\bar{\mathbf{B}}_{t-1}^{\mathrm{T}} + \frac{\beta}{\Delta t^2}\mathbf{I}\right)^{-1}\Big(\frac{\mathbf{z}_T - \mathbf{z}_t}{\Delta t} - \bar{\mathbf{A}}_{t-1}\mathbf{z}_t \\ - \bar{\mathbf{c}}_{t-1}\Big)\end{aligned} \tag{19}$$

The optimal control formulation in Eq. 12 and 18 are similar to those of Linear Quadratic Regulator (LQR) with (finite) horizon equal to 1. Detailed derivations of Eq. 13 and 19 can be seen in [22].

From Eq. 13, 14, and 19, in general we can write:

$$\mathbf{a}_{t,ID} = \mathbf{f}_{id}(\mathbf{z}_T, \mathbf{z}_t, \mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \Delta t) \tag{20}$$

For our purpose, it is mostly important that the inferred inverse dynamics action points to the right direction. Therefore, we can leverage the demonstration dataset to also optimize the following inverse dynamics loss:

$$L_{ID} = \sum_{t=1}^{H}\left\|\frac{\mathbf{f}_{id}(\mathbf{z}_T = \mathbf{z}_{t+1}, \mathbf{z}_t, \mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \Delta t)}{\|\mathbf{f}_{id}(\mathbf{z}_T = \mathbf{z}_{t+1}, \mathbf{z}_t, \mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \Delta t)\|} - \frac{\mathbf{a}_t}{\|\mathbf{a}_t\|}\right\|^2 \tag{21}$$

We combine the loss functions as follows:

$$L_{totalAE} = w_{AER}L_{AER} + w_{MDS}L_{MDS} + w_{CDP}L_{CDP} \tag{22}$$

$$L_{totalDyn} = w_{LFD}L_{LFD} + w_{ID}L_{ID} \tag{23}$$

with the weights $w_{MDS}, w_{CDP}, w_{AER}, w_{LFD}, w_{ID}$ are tuned so that each loss function components become comparable to each other. Some individual loss functions are depicted in Figure 2. $L_{totalAE}$ and $L_{totalDyn}$ are minimized separately (with separate optimizer) in parallel with respect to the human demonstrations' trajectory dataset $\{(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})\}_{t\in\{1,\ldots,H\}}$.

[5]These Jacobians exist in our experiment because we choose smooth activation functions for $\mathbf{h}_{fcnnNL}$, such as hyperbolic tangent (tanh).

IV. Experiments

A. Experimental Setup

We use the right arm of a bi-manual anthropomorphic robot system, which is a 7-degrees-of-freedom Barrett WAM arm, plus its hand which has three fingers. We mount a biomimetic tactile sensor BioTac [3] on the tip of the middle finger of the right hand. The finger joints configuration are programmed to be fixed during demonstration and testing. The setup is pictured in Figure 1. We setup the end-effector frame to coincide with the BioTac finger frame as described in [20], figure 4. The BioTac has 19 electrodes distributed on the skin surface, capable of measuring deformation of the skin by measuring the change of impedance when the conductive fluid underneath the skin is being compressed or deformed due to a contact with an object. In our experiments, $\mathbf{s}$ is a vector of 19 values corresponding to the digital reading of the 19 electrodes, subtracted with its offset value estimated when the finger is in the air and does not make any contact with any object. The contact pressure information p is a scalar quantity, which is obtained by negating the mean of the vector $\mathbf{s}$, i.e. $p = -\bar{\mathbf{s}} = -\frac{1}{19}\sum_{i=1}^{19} s_i$, with s_i being the digital reading of the i-th electrode minus its offset.

1) Human Demonstration Collection: For collecting human demonstrations, we set the robot to be in a gravity compensation mode, allowing a human demonstrator to guide the robot to a sequence of contact interaction between the BioTac finger and a drawer handle. The robot's sampling and control frequency is 300 Hz, while the tactile information $\mathbf{s}$ is sampled at 100 Hz. Later p can be computed from $\mathbf{s}$.

The demonstrations are split into two parts: one part corresponds to the contact interaction dynamics due to the rotational change of the finger pose, and the other part due to the translational change of the finger pose. Each part comprises of 7 sub-parts which correspond to contact point trajectories that traverse through different areas of the skin. For each sub-part of the rotational motion, we provide 3 demonstrations, while for the translational motion, we provide 4 demonstrations. These are determined such that we have a 50%-50% composition of data points for rotational and translational motion, respectively. Each rotational demonstration involves the sequence of making contact, rotational motion clock-wise w.r.t. x-axis of the finger frame, breaking contact, making contact again, rotational motion counter-clock-wise, and finally breaking contact. Each translational demonstration involves the sequence of making contact, swiping motion forward on the x-axis of the finger frame, breaking contact, making contact again, swiping motion backward, breaking contact again, and then repeat the whole sequence one more time. The breaking and making contacts are intentionally done to make data segmentation easier, by using a zero-crossing algorithm [23].

In total we collect $N = 55431$ data points of the tactile sensing vector $\mathbf{s}$, and $H = 183825$ pairs of $(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$. Instead of constructing a single

massive geodesic distance matrix of size $N \times N$, we split the data randomly into P bins, each of size $N' = 2310$, so we end up with P geodesic distance matrices, each of size $N' \times N'$. During training, for each siamese pair being picked, both data points must be associated with the same geodesic distance matrix. For the geodesic distance computation, we use a nearest-neighborhood of size $M = 18$. On the other hand, we obtain the number of state-action pairs H after excluding the pairs that contain states which correspond to contact pressure information p below a specific threshold. We exclude these pairs as we deem them being off-contact tactile states and not being informative for performing tactile servoing[6].

After collecting the demonstrations, we pre-process the data by performing low-pass filtering with a cut-off frequency of 1 Hz. We determined this cut-off frequency by visualizing the frequency-domain analysis of the data. This frequency selection of tactile servoing is also supported by a previous work by Johansson et al. [24]. During training, we perform the forward dynamics prediction at $\frac{100}{29}$, $\frac{100}{30}$, $\frac{100}{31}$, $\frac{100}{32}$, and $\frac{100}{33}$ Hz, while during testing, the model predict at $100/31$ Hz. In general it is hard to predict at higher frequencies, because demonstrations are performed slowly.

2) Action Representation: We choose the end-effector velocity expressed with respect to the end-effector frame as the action/policy representation. By representing the end-effector velocity with respect to the end-effector frame, effectively we are cancelling out the dependency of the state representation on the end-effector pose information, making the learned policy easier to generalize to new situations. Moreover, this choice also naturally takes care of repeatable position tracking error of the end-effector.

We use the Simulation Lab (SL) robot control framework [25] in our experiments[7]. The framework provides us with the end-effector velocity with respect to the robot base frame $\dot{\mathbf{x}}_b$. To get the end-effector velocity with respect to the end-effector frame $\dot{\mathbf{x}}_e$, we compute the following [27]:

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \mathbf{R}_e^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_e^{\mathrm{T}} \end{bmatrix} \dot{\mathbf{x}}_b \tag{24}$$

where $\mathbf{R}_e$ is the end-effector orientation with respect to the base frame, expressed as a rotation matrix. Hence, we define the action $\mathbf{a} = \dot{\mathbf{x}}_e$ with dimensionality 6, where the first three dimensions is the linear velocity and the last three is the angular velocity. During the demonstration, the robot is sampled at 300 Hz, but the prediction is made at 3 Hz, for this we summarize by averaging all $\dot{\mathbf{x}}_b$'s applied between $\mathbf{s}_t$ and $\mathbf{s}_{t+1}$, and then convert this average to $\dot{\mathbf{x}}_e$.

3) Machine Learning Framework and Training Process: Our auto-encoder takes in 19 dimensional input vector $\mathbf{s}$, and compresses it down to a 3 dimensional latent state embedding, $\mathbf{z}$. The intermediate hidden layers are fully connected layers of size 19, 12, 6, all with *tanh* activation function, forming the encoder function $\mathbf{f}_{enc}$. The decoder part is a mirrored structure of the encoder function, forming $\mathbf{f}_{dec}$. $\mathbf{h}_{fcnnNL}$ is a feedforward neural network with 9 dimensional input (3 dimensional latent state $\mathbf{z}$ and 6 dimensional action policy $\mathbf{a}$), 1 hidden layer of size 15 with tanh activation functions, and 3 dimensional output. $\mathbf{h}_{fcnnLL}$ is a feedforward neural network with 3 dimensional latent state $\mathbf{z}$ as input, 3 hidden layers of size 8, 15, 23, all with tanh activation function, and 30 dimensional output which corresponds to the parameters of $\mathbf{A}_t$, $\mathbf{B}_t$, and $\mathbf{c}_t$ in Eq. 9.

We use a batch size of 128, and we use separate RMSProp optimizers [28] to minimize $L_{totalAE}$ and $L_{totalDyn}$ for 200k iterations. We set the values of $w_{MDS} = 2 \times 10^7$, $w_{CDP} = 2 \times 10^7$, $w_{AER} = 100$, $w_{LFD} = 1 \times 10^8$, $w_{ID} = 1 \times 10^7$, and $\beta = 0.1$ empirically. We implement all components of our model in TensorFlow [29]. We also noticed a significant improvement in learning speed and fitting quality after we add Batch Normalization [30] layers in our model.

B. Auto-Encoder Reconstruction Performance

Our first evaluation is on the reconstruction performance of the auto-encoder in terms of normalized mean squared error (NMSE). NMSE is the mean squared prediction error divided by the variance of the ground truth. We obtain all NMSE values are below 0.25 for the training (85% split), validation (7.5% split), and test (7.5% split) sets.

C. Neural Network Multi-Dimensional Scaling (MDS)

In terms of MDS performance, we plot the x-y coordinates of the latent space embedding of all tactile sensing $\mathbf{s}$ data points in the demonstration data, in Figure 3. Each data point is colored and labeled based on the BioTac electrode index with maximum activation. This result agrees with the Figure 2 of [20]. Moreover, we randomly sampled 10000

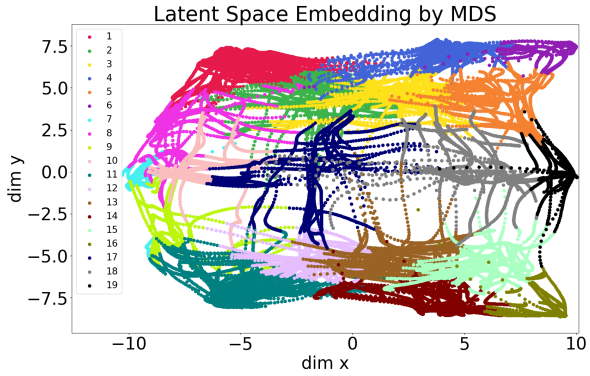


Fig. 3. x-y dimensions of latent space embedding by MDS

siamese pairs from the training, validation, and test dataset, and compare their x-y Euclidean distance in the latent space vs. the ground truth geodesic distance. We got all these comparisons to have NMSE less than 0.02.

D. Latent Forward Dynamics Prediction Performance

We trained the latent space forward dynamics function $\mathbf{f}_{fd}$ by chain-predicting the next C latent states with a length of training chain $C_{train} = 2$ and testing it with a length of chain $C_{test} = 3$. We then evaluate the NMSEs. In Figure 4, we compare the performance between 4 different combinations:

- using both of L_{MDS} and L_{CDP} loss functions during training as indicated by *LatStruct* or not using both of them –i.e. without any structure imposed in the latent space representation– as indicated by *noLatStruct*, and

[6]In the extreme case, when the robot is not in contact with any object, there is no point of performing tactile servoing.

[7]In the previous version of our experiment we use Riemannian Motion Policies (RMP) [26] for the robot control framework.

- using inverse dynamics loss L_{ID} during training as indicated by *IDloss* or not using it (*noIDloss*).

We see that in all cases where no latent space structure is imposed, the performance is generally worse than those with imposed latent space structure. We believe this happens because it is a hard task to train a forward dynamics predictor to predict on an unstructured latent space. On the other hand, in general we see that all models with imposed inverse dynamics loss L_{ID} perform worse than those without L_{ID}. We think this is most likely because training a model without imposing L_{ID} loss is easier than training with imposing it. However, as we will see in section IV-E, the model trained without L_{ID} loss does not provide correct action policies for tactile servoing as it was not trained to do so.

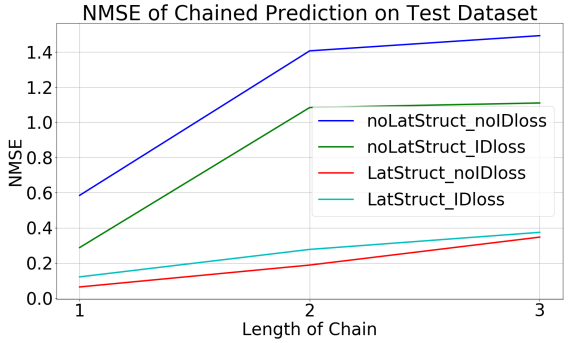


Fig. 4. Normalized mean squared error (NMSE) vs. the length of chained forward dynamics prediction, averaged over latent space dimensions, on test dataset.

E. Inverse Dynamics Prediction Performance

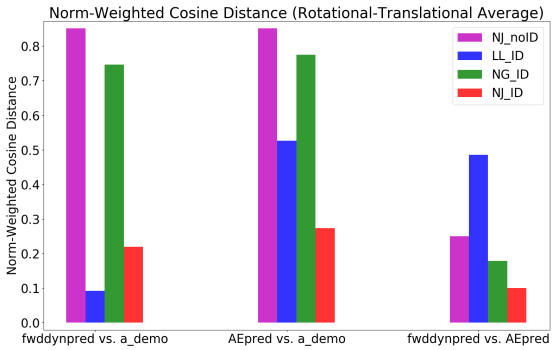
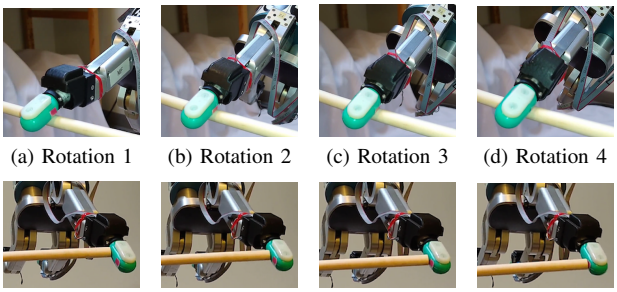


Fig. 5. Average cosine distance between rotational and translational inverse dynamics, weighted by the norm of the ground truth.

In Figure 5, we compare the inverse dynamics prediction performance between the 3 possible inverse dynamics models as described in section III-E, in terms of the average cosine distance between rotational and translational[8] inverse dynamics, weighted by the norm of the ground truth. We termed *fwddynpred* and *AEpred* for inverse dynamics prediction in Eq. 20 by setting $\mathbf{z}_T$ equal to the *constant* value of $\mathbf{f}_{dfd}(\mathbf{z}_t, \mathbf{a}_t)$ and $\mathbf{f}_{enc}(\mathbf{s}_{t+1})$, respectively. Obviously $\mathbf{f}_{dfd}(\mathbf{z}_t, \mathbf{a}_t)$ is an easier target for inverse dynamics than $\mathbf{f}_{enc}(\mathbf{s}_{t+1})$, as apparent from the better prediction performance of the left bar group as compared to the middle bar group. If $\mathbf{f}_{fd}$ is trained well, we can expect that the performance between *fwddynpred* and *AEpred* becomes more similar. On the right bar group, we also compare *fwddynpred* vs. *AEpred*: poor performance here indicate whether $\mathbf{f}_{fd}$ could not predict well, or unstable $\mathbf{f}_{id}$ (i.e. a big change in $\mathbf{a}_{t,ID}$ for a small change in $\mathbf{z}_T$). With respect to this analysis, we deem Neural Network Jacobian (NJ) to be the best inverse dynamics model. We also evaluate *NJ_noID* which corresponds to not minimizing L_{ID} loss. By comparing *NJ_noID* and *NJ_ID*, we can see that minimizing L_{ID} loss is essential for acquiring a good inverse dynamics model.

F. Real Robot Experiment



(a) Rotation 1 (b) Rotation 2 (c) Rotation 3 (d) Rotation 4

(e) Translat. 1 (f) Translat. 2 (g) Translat. 3 (h) Translat. 4

Fig. 6. Snapshots of our experiments executing the tactile servoing with the learned model (non-linear LFD model and neural network Jacobian ID model) on a real robot. Red sticker indicates the target contact point. The first row, figures (a)-(d) are for a target contact point whose achievement requires rotational change of pose of the BioTac finger. The second row, figures (e)-(h) are for a target contact point whose achievement requires translational change of pose of the BioTac finger.

In Fig. 6, we provide snapshots of robot executions on a real hardware[9] with real-time tactile sensing from the BioTac finger. We see that the system is able to produce the required rotational motions (Fig. 6 (a)-(d)) and translational motions (Fig. 6 (e)-(h)) needed to achieve the specified target contact point. The full pipeline of the experiment can be seen in the video `https://youtu.be/0QK0-Vx7WkI`.

V. Discussion and Future Work

In this work, we presented a learning-from-demonstration framework for achieving tactile servoing behavior. We showed that our manifold representation learning of tactile sensing information is critical to the success of our approach. We also showed that for learning a tactile servoing model, it is important to not only be able to predict the next state from the current state and action (forward dynamics), but also be able to predict the action if given a target state (inverse dynamics).

In the future, we would like to extend our work to not only track a contact point, but also a contact profile surrounding the point. This can be useful to produce interesting behaviors such as tactile navigation on the edges of an object.

Acknowledgment

We thank David Crombecque and Ragesh Kumar Ramachandran, both from the University of Southern California, for the insightful discussions on mathematical manifolds. We also thank Arunkumar Byravan for discussions on the SE3-Pose-Nets paper, and Kendall Lowrey for the help on a finishing work of the BioTac mounting for an earlier version of the work, both from the University of Washington.

[8]Translational and rotational components here correspond to the first three and the last three dimensions of $\mathbf{a}$, respectively.

[9]The model gives $\dot{\mathbf{x}}_e$ as output, while the robot only knows how to track $\dot{\mathbf{x}}_b$, thus we need to invert Eq. 24 to perform tactile servoing.

REFERENCES

[1] R. Johansson, "Light a match: Normal, pre-anesthetization performance vs post-anesthetization performance," https://www.youtube.com/watch?v=0LfJ3M3Kn80, 2018, accessed: 2018-08-04.

[2] R. S. Johansson and G. Westling, "Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects," *Experimental Brain Research*, vol. 56, no. 3, pp. 550–564, Oct 1984. [Online]. Available: https://doi.org/10.1007/BF00237997

[3] N. Wettels, V. Santos, R. Johansson, and G. Loeb, "Biomimetic tactile sensor array." *Advanced Robotics*, vol. 22, no. 8, pp. 829–849, 2008.

[4] C. Chorley, C. Melhuish, T. Pipe, and J. Rossiter, "Development of a tactile sensor based on biologically inspired edge encoding," in *2009 International Conference on Advanced Robotics*, June 2009, pp. 1–6.

[5] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/12/2762

[6] P. Mittendorfer and G. Cheng, "Humanoid multimodal tactile-sensing modules," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 401–410, June 2011.

[7] Q. Li, C. Schürmann, R. Haschke, and H. J. Ritter, "A control framework for tactile servoing," in *Robotics: Science and Systems*, 2013.

[8] N. F. Lepora, K. Aquilina, and L. Cramphorn, "Exploratory tactile servoing with active touch," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1156–1163, April 2017.

[9] A. Byravan, F. Leeb, F. Meier, and D. Fox, "Se3-pose-nets: Structured deep dynamics models for visuomotor planning and control," *CoRR*, vol. abs/1710.00489, 2017. [Online]. Available: http://arxiv.org/abs/1710.00489

[10] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 2746–2754. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969442.2969546

[11] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," *CoRR*, vol. abs/1606.07419, 2016. [Online]. Available: http://arxiv.org/abs/1606.07419

[12] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, ser. CVPR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 1735–1742. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2006.100

[13] G. Pai, R. Talmon, and R. Kimmel, "Parametric manifold learning via sparse multidimensional scaling," *CoRR*, vol. abs/1711.06011, 2017. [Online]. Available: http://arxiv.org/abs/1711.06011

[14] Z. Su, J. Fishel, T. Yamamoto, and G. Loeb, "Use of tactile feedback to control exploratory movements to characterize object compliance," *Frontiers in neurorobotics*, vol. 6, p. 7, 07 2012.

[15] H. van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 121–127.

[16] J. Sung, J. K. Salisbury, and A. Saxena, "Learning to represent haptic feedback for partially-observable tasks," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 2802–2809.

[17] V. Kumar, A. Gupta, E. Todorov, and S. Levine, "Learning dexterous manipulation policies from experience and imitation," *CoRR*, vol. abs/1611.05095, 2016.

[18] G. Sutanto, Z. Su, S. Schaal, and F. Meier, "Learning sensor feedback models from demonstrations via phase-modulated neural networks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1142–1149.

[19] G. Pai, R. Talmon, and R. Kimmel, "Parametric manifold learning via sparse multidimensional scaling," *CoRR*, vol. abs/1711.06011, 2017. [Online]. Available: http://arxiv.org/abs/1711.06011

[20] C.-H. Lin, J. A. Fishel, and G. E. Loeb, "Estimating point of contact , force and torque in a biomimetic tactile sensor with deformable skin," in *SynTouch LLC*, 2013.

[21] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," *CoRR*, vol. abs/1708.02596, 2017.

[22] G. Sutanto, N. D. Ratliff, B. Sundaralingam, Y. Chebotar, Z. Su, A. Handa, and D. Fox, "Learning latent space dynamics for tactile servoing," *CoRR*, vol. abs/1811.03704, 2018. [Online]. Available: http://arxiv.org/abs/1811.03704

[23] A. Fod, M. J. Matarić, and O. C. Jenkins, "Automated derivation of primitives for movement classification," *Autonomous robots*, vol. 12, no. 1, pp. 39–54, 2002.

[24] R. Johansson and J. Flanagan, "Coding and use of tactile signals from the fingertips in object manipulation tasks," *Nature reviews. Neuroscience*, vol. 10, pp. 345–59, 05 2009.

[25] S. Schaal, "The sl simulation and real-time control software package," University of Southern California, Los Angeles, CA, Tech. Rep., 2009, clmc. [Online]. Available: http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf

[26] N. D. Ratliff, J. Issac, and D. Kappler, "Riemannian motion policies," *CoRR*, vol. abs/1801.02854, 2018.

[27] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[28] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Man, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Vigas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2015. [Online]. Available: http://download.tensorflow.org/paper/whitepaper2015.pdf

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.