# Self-Supervised Learning for Specified Latent Representation

Chicheng Liu, Libin Song, Jiwen Zhang, Ken Chen, and Jing Xu

***Abstract*—Current latent representation methods using unsupervised learning have no semantic meaning; thus, it is difficult to directly express their physical task in the real world. To this end, this paper attempts to propose a specified latent representation with physical semantic meaning. First, a few labeled samples are used to generate the framework of the latent space, and these labeled samples are mapped to framework nodes in the latent space. Second, a self-learning method using structured unlabeled samples is proposed to shape the free space between the framework nodes in the latent space. The proposed specified latent representation therefore possesses the advantages provided by both supervised and unsupervised learning. The proposed method is verified by numerical simulations and real-world experiments.**

***Index Terms*—Latent representation, neural networks, unsupervised learning.**

## I. Introduction

THERE are only three degrees of freedom (DOF) of an object rotating in 3D space. However, if the rotation process is recoded by a camera with a series of images, then the low-dimensional rotation is represented by the high-dimensional images since each image can be regarded as a high-dimensional vector. In other words, these high-dimensional images contain a latent representation of three-dimensional rotation. There are many similar problems; for example, a series of images of a car traveling from near to far contain one-dimensional latent representations of distances, a sequence of photographs of a person at different ages contains a one-dimensional latent representation of age, and a series of images of a 6-DOF robotic manipulator contains a six-dimensional latent representation of joint angles. If we can obtain the physical semantic latent representation of such images, many tasks can be easily performed. Taking visual servoing for example, if we can obtain the latent representation of the relative pose between an object and a camera, we can directly control the motions of the camera or the object.

Using neural networks to learn the latent representations of images has been widely studied in recent years [1]–[4]. Within the community of latent representation learning, there two broad parallel lines of methods that have developed: one line rooted in probabilistic models and one line rooted in neural networks. Both lines represent unsupervised learning methods, whose significant advantage is that only unlabeled samples are needed, as opposed to labeled samples. In particular, the restricted Boltzmann machine [5]–[7] is mainly used on the probabilistic side, and auto-encoder variants [8]–[10] are used on the neural network side. However, the significant shortcoming of these unsupervised learning models is that they are not focused on learning latent representations that have physical semantic meanings. Thus, the latent representations cannot directly represent the model's performance in the real world; that is, the performance described in the first paragraph cannot be represented by these methods alone.

If a large number of labeled samples can be obtained, we can use supervised learning methods to directly learn a specified latent representation with physical semantic meaning [11]. Thus, the learning of specified latent representations can be regarded as a regression problem, which has been widely studied. However, it is difficult to obtain the label of each sample in some cases since the acquisition of labels requires massive measurements, expensive equipment, or special experimental environments.

In this paper, we present a method that can learn a specified latent representation with a few labeled samples and a large number of unlabeled samples, therein combining the advantages of unsupervised learning and supervised learning methods. The specified latent representation is one of the many feasible forms that can represent the real world. For example, the specified latent representation of images of a rotating object in this paper is three Euler angles that can be used to describe the orientation of objects in the world coordinate system. For this purpose, first, a few labeled samples are used to construct the regression loss to build up the framework of the latent space, and labeled samples are mapped to framework nodes in the latent space. Unfortunately, the prediction accuracy of the free space between framework nodes is low due to underfitting of supervised learning caused by insufficient data. Second, a large number of structured unlabeled samples are used to address the structure loss to further shape the free space between framework nodes to improve the prediction accuracy in the free space.

Manuscript received June 6, 2018; revised December 28, 2018, February 22, 2019, and March 1, 2019; accepted March 5, 2019. Date of publication March 11, 2019; date of current version January 2, 2020. This work was supported in part by the National Key R&D Program of China under Grant 2017YFC0822204, in part by the National Natural Science Foundation of China (NSFC) under Grants U1613205 and 51675291, and in part by the State Key Laboratory of China under Grant SKLT2018C04. *(Corresponding author: Jing Xu.)*

The authors are with the State Key Laboratory of Tribology, the Beijing Key Laboratory of Precision/Ultra-Precision Manufacturing Equipment Control, and the Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China (e-mail: liucc13@mails.tsinghua.edu.cn; songlb@mail.tsinghua.edu.cn; jwzhang@tsinghua.edu.cn; kenchen@tsinghua.edu.cn; jingxu@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2019.2904237

The structured unlabeled samples are specially designed to shape the free space between the framework nodes. The free space between the framework nodes can be regarded as twisted space details that are not flattened by the framework generated by supervised learning with the labeled samples. The structured unlabeled samples used in this paper are grouped together, and the unlabeled samples in the same sequence are equally distributed in the latent space. It is easy to obtain such structured unlabeled samples in some cases. For example, if a car is moving away from the camera at a constant speed, the frames in the captured video can be regarded as a sequence of structured unlabeled samples because the distances between the cars in all adjacent frames are equal. The structure information in each sequence of structured unlabeled samples, which can flatten these twisted space details, are used to construct the structure loss to shape the free space; thus, the prediction accuracy of the latent representation in this free space can be improved. Since the structure information of each sequence is used by itself, it is called self-supervised learning in this paper.

In this paper, our proposed method is embedded with different types of networks, and their performances are compared. The generality of our proposed method is also verified by some typical applications.

Our contributions are as follows:

1) A structure loss is proposed for structured unlabeled samples for self-supervised learning, which provides more information than traditional unlabeled samples for unsupervised learning and greatly reduces the number of labeled samples.
2) Supervised learning and a proposed self-supervised learning are combined for the learning of a specified latent representation.
3) The mechanism of the proposed method is studied, and the factors influencing the prediction accuracy are analyzed.

Our code is available at https://github.com/liucc09/self-supervised. Your can check out more results there.

## II. Related Work

*Learning Representation:* Representation learning is a field in the machine learning community, and representation learning has many successful applications in both academia and industry such as speech recognition and processing [12]–[17], object recognition [18]–[21], natural language processing [22]–[25], and transfer learning [26]–[29]. The question of representation learning can be interpreted as an attempt to recover the latent random variables that describe a distribution over observed data. For some tasks, we have prior knowledge of the latent representation, and we want the networks to learn a specified latent representation instead of any feasible representation. In this paper, we focus on how to learn a specified latent representation.

*Autoencoder:* The latent representation of high-dimensional data can be learned by training a multilayer autoencoder [4], [30] with a small central layer to reconstruct high-dimensional input vectors. Some studies use an autoencoder to learn a low-dimensional latent representation of input images and then perform specific tasks based on the latent representation. Byravan *et al.* [2] design SE3-Pose-NETS, which learns a low-dimensional pose embedding for visuomotor control via an autoencoder structure. The low-dimensional pose is used as the input to a three-layer network that is used to predict the 6D pose. Watter *et al.* [1] also build their embedded control method based on autoencoders, and the locally linear dynamics are estimated and used to control the motion of an agent. Finn *et al.* [4] present an approach that can learn a state representation based on a deep spatial autoencoder, and reinforcement learning is used based on the state representation.

However, in contrast to these prior works, our approach attempts to learn a specified low-dimensional latent representation that has physical semantic meaning.

*Direct Encoding:* Several other methods directly learn the encoding of the high-dimensional inputs based on supervised learning. Bateux *et al.* [11] present a deep neural network-based method that can directly learn the 6DOF relative pose from raw images. Semi-supervised embedding [31] uses unsupervised dimensionality reduction algorithm before learning direct encoding. Several approaches [7], [32], [33] also use semi-supervised methods to learn a parametric mapping based on a neighborhood graph. Supervised methods can be regarded as learning direct encoding, but these methods need a large number of labeled samples. The semi-supervised methods in the literature mainly use unsupervised methods to learn a better representation of the inputs, and then use supervised methods to learn the encoding based on the learned representation.

However, in contrast to the above-mentioned approaches, our method attempts to use the proposed self-supervised learning technique to reduce the number of labeled samples.

## III. Method

This paper attempts to learn mapping functions from a raw image $X$ to a specified latent representation $Z$ (such as object orientation, location, and pose). To format the framework of the latent space for the semantics and reduce the number of labeled samples, supervised learning and the proposed self-supervised learning technique are combined in this paper. Our objective function contains two types of terms: the *regression losses* and the *structure losses*. *Regression losses* are used for supervised learning and can be used to build up the framework of the latent space. *Structure losses* are used for the self-supervised learning and use unlabeled samples to shape the free space between the framework nodes generated by the previous supervised learning.

### A. Regression Loss

The regression losses are used to build up the framework of the latent space by minimizing the prediction errors of the labeled samples. For the mapping function $G : X \to Z$, we express the regression loss $\mathscr{L}_{\mathrm{re}}$ as

$$\mathscr{L}_{\mathrm{re}}\left(X_i, Z_i\right) = \left\|G\left(X_i\right) - Z_i\right\|^2 \tag{1}$$

where the subscript $i$ is the sample index.

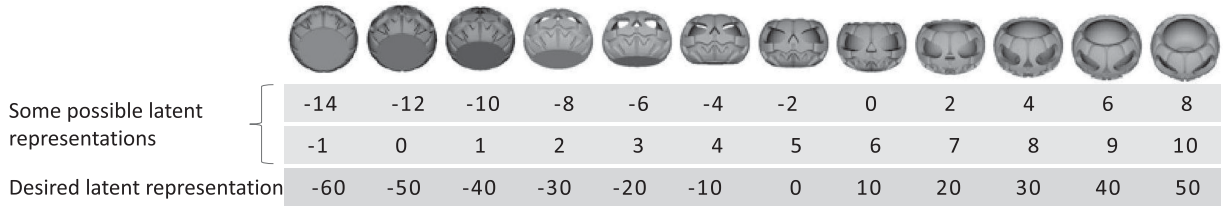



Fig. 1. Possible latent representations. The sequence of pumpkin pictures is created by rotating a 3D pumpkin model from top to bottom. The specified latent representation is the rotation angles about the horizontal axis.

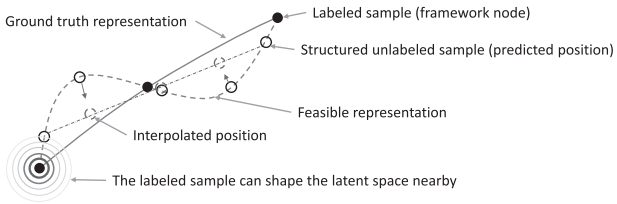



Fig. 2. Illustration of the structure loss. The labeled samples are used to determine the general framework of the latent space, and the unlabeled structure samples are used to shape the free space between the framework nodes.

The regression loss can be used to define the framework of the latent space with supervised learning using labeled samples, which will further determine the specified latent representation. Meanwhile, the labeled samples are mapped into frameworks nodes in the latent space. As shown in Fig. 1, the pictures are generated by rotating a 3D pumpkin model from top to bottom about the horizontal axis. Therefore, these pictures are one-dimensional manifolds in a high-dimensional space. The latent representation can be a series of vectors in an arbitrary one-dimensional manifold. Therefore, there are infinite possible latent representations with different frameworks of the latent space such as the first two rows in Fig. 1. However, the latent representation that we want is that representation with physical semantic meaning, which represents the actual rotation angle about the horizontal axis as the third row in Fig. 1. To this end, the regression loss is designed to build up the framework of the latent space (such as the rotation angle about the horizontal axis, as shown in Fig. 1).

### *B. Structure Loss*

Due to the insufficient labeled samples for the above supervised learning, the prediction accuracy of the free space between framework nodes is low, that is, the free space between the framework nodes is unconstrained. To this end, the structure loss is used to shape the free space between the framework nodes generated by the previous supervised learning. As shown in Fig. 2, the solid spots represent the ground truth of the labeled samples, which are used to generate the framework of the latent space via supervised learning. Therefore, the solid black spots are also framework nodes in the latent space. However, the free space between the framework nodes remains unconstrained. To shape the free space between the framework nodes, the structured unlabeled sample is employed. There is a sequence of unlabeled samples with known relationships between unlabeled samples in the same sequence; for example, this sequence of unlabeled samples is evenly distributed in the latent space. The predicted positions of this sequence of unlabeled samples in the latent space, shown as a solid circle in Fig. 2, are generated by the network trained by labeled samples using supervised learning. Assuming that the first and last unlabeled samples are perhaps closer to the framework nodes than the other unlabeled samples in this sequence, the predicted positions of the first and last unlabeled samples are more accurate than the predicted positions of the other unlabeled samples. In this paper, the positions (called interpolated positions) of the other unlabeled samples (except the first and last unlabeled samples) in this sequence in the latent space are also linearly interpolated by the predicted positions of the first and last unlabeled samples, as shown by the dashed circle in Fig. 2. Obviously, the predicted position and interpolated position of the other unlabeled samples are not coincident. Predicted position is regarded as target position in each epoch. To improve the position accuracy and shape the free space, the distance between the target position and interpolated position in the latent space is minimized by the structure loss $\mathscr{L}_{\mathrm{st}}$ as

$$\mathscr{L}_{\mathrm{st}}\left(Y_i^j\right) = \left\| G\left(Y_i^j\right) - F\left(Y_i^j | Y_1^j, \ldots, Y_{n_j}^j\right)\right\|^2 \quad (2)$$

where superscript $j$ is the sequence index, $Y_i^j$ represents the $i$th unlabeled samples in the $j$th sequence, $F$ is a function that gives the interpolations of $Y_i^j$ based on the structure information of the sequence, and $n_j$ is the total number of the samples in the $j$th sequence.

### *C. Full Objective*

Our full objective function is

$$\mathscr{L}_{\mathrm{full}} = \sum_{i=1}^{m_{\mathrm{re}}} \mathscr{L}_{\mathrm{re}}\left(X_i, Z_i\right) + \lambda \sum_{j=1}^{m_{\mathrm{st}}} \sum_{i=1}^{n_j} \mathscr{L}_{\mathrm{st}}\left(Y_i^j\right) \quad (3)$$

where $m_{re}$ is the number of labeled samples, $m_{st}$ is the number of sequences of structured unlabeled samples, and $\lambda$ controls the weight of the two objectives. We aim to solve the following:

$$G^* = \arg\min_G \mathscr{L}_{\mathrm{full}}\left(G\right). \quad (4)$$

Fig. 3 illustrates the mechanism of the proposed method in the 2D latent space. The red dots and green dots represent the latent representations of the labeled samples and structured unlabeled samples for training, respectively. For clear illustration,

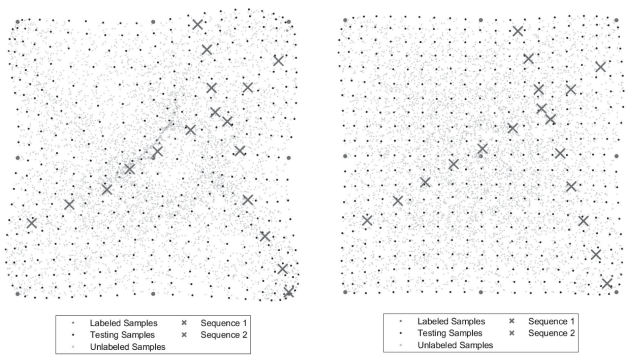



Fig. 3. Illustration of the mechanism of the proposed method. (a) Latent representations after training with only the labeled samples using the regression loss. (b) Latent representations after training with both labeled samples and structured unlabeled samples using the full objective.

two sequences of structured unlabeled samples are selected and marked with crosses. Black dots represent the latent representations of the testing samples. The desired distributions of the latent representations of the labeled samples and the testing samples are grid distributions. As shown in Fig. 3(a), after training with only labeled samples using the regression loss, the latent representations of the labeled samples are accurate while the latent representations of the testing samples are not accurate. The latent space between the labeled samples is twisted. To flatten the twisted latent space, we take advantage of the structure information of the structured unlabeled samples. The structure information is that the latent representations of the unlabeled samples in the same sequence should be equally spaced along a straight line. Fig. 3(b) shows the results after 20 epochs of training with both labeled samples and structured unlabeled samples using the full objective. We can see that the latent representations of the structured unlabeled samples in each sequence are equally spaced along a straight line as desired; each line can flatten the latent space where it passes through. As the number of sequences of the unlabeled samples becomes larger, the number of straight lines becomes larger. The twisted latent space is flattened by more straight lines; thus, the latent representations of the testing samples improves.

Notice that our model is not based on a specific network architecture; it can be embedded in a wide range of networks. In Section IV, we embed our model in different networks and compare their performance.

### D. Data Preprocessing

This paper attempts to learn a continuous latent representation, in contrast to classification whereby discrete labels are used to represent different classes. In this paper, if raw images are directly translated into discrete image vectors, then the discrete image vectors are further mapped to a latent representation (such as rotation angle). Specifically, as illustrated in Fig. 4, a dark spot moves from left to right in a series of one-dimensional images in the top, and one of the possible latent representations

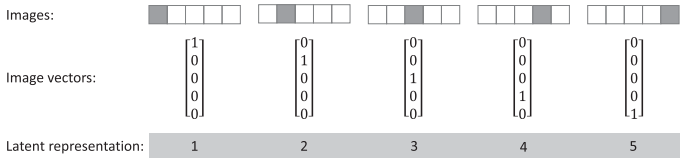



Fig. 4. Illustration of the discrete image vectors and the continuous latent representation.

is shown at the bottom. However, the variations in each dimension of the image vectors with respect to the latent representation are not continuous. Because of dimension curse, learning continuous outputs from discrete inputs is more difficult than with continuous inputs.

In this case, principal component analysis (PCA) [34] is used to transform the discrete image vectors into continuous image vectors to generate continuous inputs. In addition, PCA is also able to reduce the dimensions of the image vectors. Specifically, we perform PCA on all the training samples as shown in (5). To reduce the dimensions of the image vectors $Vs$ after transformation by PCA, we only retain the dimensions of the image vectors $Vs$ if the corresponding importances of each principal component $Ws$ is greater than $0.01\%$ of the greatest importance. Therefore, images of $50 \times 50$ pixels can be translated to column vectors of approximately 40 dimensions.

$$[Ps, Vs, Ws] = \text{PCA}(\text{images}) \tag{5}$$

where $Ps$ represent the principal components.

In this paper, PCA is only used to preprocess the raw images for networks without convolutional layers. For convolutional neural networks (CNNs), the convolutional layers and the pooling layers can avoid the problems caused by discretization, thus, PCA is not required.

The next step, following preprocessing, is normalization, which normalizes all the elements of the image vectors to $[-1, 1]$. For the input, all the elements in every dimension in the input vectors are normalized as a whole, whereas for the output, the elements in the same dimension in the output vectors are normalized individually.

### E. Implementation

When our proposed method is used in network architectures, the proposed structure loss and training method result in training processes that are different from the traditional methods. Specifically, in (2), the target in the training is dynamically updated in each epoch. The flowchart of each epoch is shown in Fig. 5. First, the predicted positions of all the labeled samples and structured unlabeled samples in the latent space are predicted by the current network. Second, the predicted positions of the first and last unlabeled samples and the structural information are combined to linearly interpolate the positions of other unlabeled samples in the latent space. Third, the ground truth positions of labeled samples and the interpolated positions of unlabeled samples are concatenated as the target positions of the labeled samples and the unlabeled samples. Fourth, as every sample used in the training has a target position at this step, a

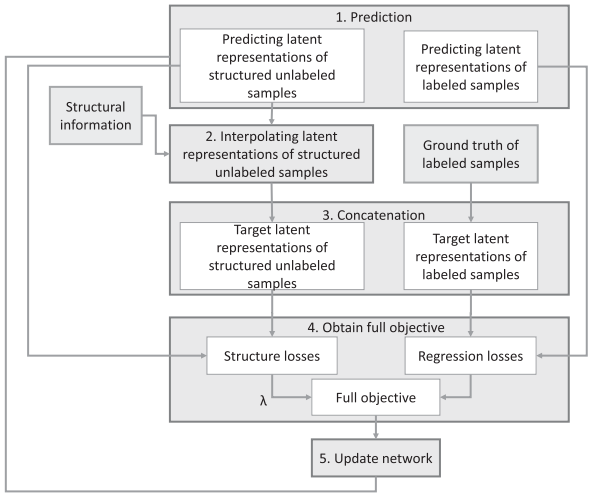



Fig. 5. Flowchart of every epoch.

traditional regression method with mean squared error (MSE) is used, the MSEs of labeled samples are regarded as the regression loss and the MSEs of unlabeled samples are regarded as the structure loss. Therefore, the full objective is obtained by combining the regression loss and the structure loss with weight $\lambda$. Finally, the network is updated by the gradient descent method and the updated network is used in the next epoch.

## IV. Results

First, our method is embedded in several network architectures, and their performances are compared on simulated datasets, where samples are generated by observing a rotating 3D pumpkin model. Second, the weight of the structure loss and the impact of the spatial distribution of the labeled samples are studied, and our full algorithm is compared against several variants. Third, our method is evaluated on a public dataset. Finally, the generality of our method is demonstrated on some applications whereby labeled data are difficult to obtain.

### A. Evaluation

The same evaluation datasets and metrics are used to test the performance of our method in several network architectures. The task is to learn a specified latent representation from raw images with a small number of labeled samples and a large number of unlabeled samples in sequences. An ablation study on the full algorithm is also performed.

*1) Datasets:* The datasets are captured images of a rotating 3D pumpkin model. The rotations are performed in 3D space; thus, the latent representation is a series of three-dimensional vectors. The 3D pumpkin model is the same as the model illustrated in Fig. 1. The resolution of the images is $54 \times 54$. The datasets consist of three parts: the labeled samples, the structured unlabeled samples, and the test samples. The structured unlabeled samples are divided into sequences. The images in a sequence are evenly distributed in the latent space, and the images are arranged in order. All the samples were distributed within $60^\circ \times 60^\circ \times 60^\circ$ in 3D space. A total of 150 labeled random samples and 1000 sequences of structured unlabeled samples were generated. In addition, the samples at the beginning and end of every sequence were randomly generated, and the other samples in the sequence were generated in the same interval. The smallest interval is $5^\circ$. The total number of samples in the 1000 sequences is 8364. In addition, 1000 test samples were randomly generated.

TABLE I
Average Prediction Error of Different Network Architectures on Labeled Samples, Structured Unlabeled Samples, and Test Samples

| Approach | Loss | Labeled | Unlabeled | Test |
|---|---|---|---|---|
| 1 | FNN + PCA+ regression loss | $\mathbf{0.07}^\circ$ | $4.31^\circ$ | $4.31^\circ$ |
| 2 | CNN+regression loss | $0.32^\circ$ | $8.69^\circ$ | $9.25^\circ$ |
| 3 | FNN + PCA + full objective | $0.54^\circ$ | $\mathbf{1.36}^\circ$ | $\mathbf{1.86}^\circ$ |
| 4 | CNN + full objective | $0.89^\circ$ | $3.15^\circ$ | $4.00^\circ$ |
| 5 | FNN + full objective | $0.14^\circ$ | $1.61^\circ$ | $2.17^\circ$ |
| 6 | FNN + PCA + structure loss | diverge | diverge | diverge |

*2) Metric:* We use the MSE between the predicted latent representation and the ground truth as the evaluation metric. The MSEs of the labeled samples, unlabeled samples, and test samples are given.

*3) Comparisions:*

1) FNN + PCA + regression loss: A feed-forward neural network (FNN) [35] with one hidden layer of 100 nodes. The FNN is only trained with the regression losses. The samples are first transformed by PCA and then downsampled to 46-dimensional column vectors.
2) CNN + regression loss: A CNN with three convolutional layers and a fully connected layer. The CNN is only trained with regression losses.
3) FNN + PCA + full objective: The same network architecture and samples as the FNN + PCA, except with the full objective function.
4) CNN + full objective: The same network architecture as the CNN except with the full objective function.
5) FNN + full objective: A feed-forward neural network with one hidden layer of 100 nodes. Raw images are used as inputs, and the full objective function is used in the training process.
6) FNN + PCA + structure loss: The same network architecture as the FNN + PCA, but only the structure losses are used in the training process.

As shown in Table I, approach 3 has better performance than approach 5, which indicates that PCA preprocessing can improve the performance when using an FNN architecture. The reasons are given in Section III-D. Approach 3 is better than approach 1 and approach 4 is better than approach 2, which means that the structure loss can improve performance both with an FNN architecture and a CNN architecture. The reasons are given in Section III-C. Approach 6 cannot converge, which means that the regression loss is necessary.

### B. Analysis

*1) Relation Between the Prediction Errors and the Number of Nodes in the Hidden Layer:* It is well known that fewer nodes

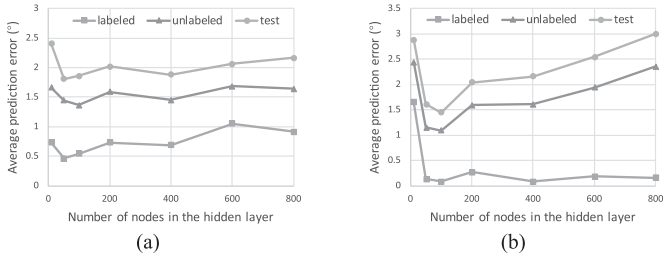



Fig. 6. Relation between the prediction errors and the number of nodes in the hidden layer. (a) One-hidden-layer FNN using the sigmoid activation function. (b) One-hidden-layer FNN using the ReLu activation function.

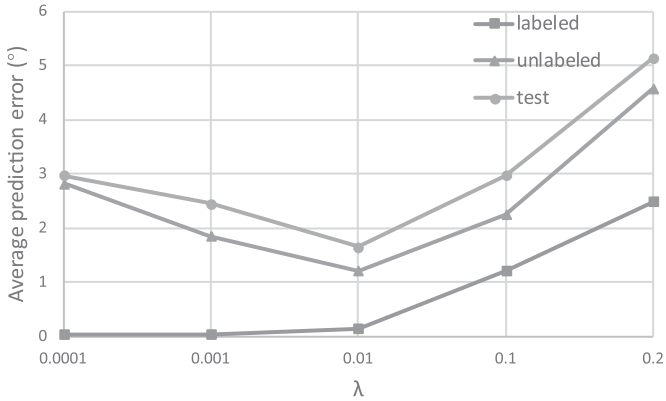



Fig. 7. Relation between the prediction error and λ.

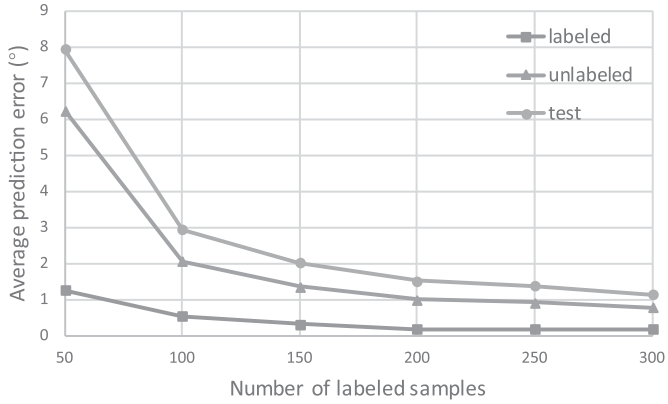



Fig. 8. Relation between prediction errors and the number of labeled samples.

in a network results in a shorter training time. However, if the number of nodes is too small, the prediction accuracy will be negatively affected. To this end, we used an FNN with only one hidden layer to study the relation between the prediction errors and the number of nodes in the hidden layer. In these experiments, the training stopped when the prediction errors of the structured unlabeled samples no longer decreases. The results are shown in Fig. 6. We train FNNs with 10, 50, 100, 200, 400, and 800 nodes in the hidden layer and use either the sigmoid or ReLu activation functions. The results for the sigmoid activation function are shown in Fig. 6(a), and the results for the ReLu activation function are shown in Fig. 6(b).

We can see that the best number of nodes in the hidden layer is approximately 50–100. The increase in the number of nodes has less influence on FNNs using the sigmoid activation function than those using the ReLu activation function. The prediction error increases significantly as the number of nodes increases in the FNNs using the ReLu activation function. This is because FNNs using the sigmoid activation function are more difficult to overfit on labeled samples. The sigmoid activation function presents more serious problems of vanishing gradients than does the ReLu activation function.

Another conclusion is that FNNs using the ReLu activation function can achieve higher accuracy than those using the sigmoid activation function if the number of nodes is properly selected.

*2) Relation Between Prediction Errors and λ:* The parameter λ is used to adjust the relative weight of the labeled samples and the structured unlabeled samples. A larger λ results in more weight added to the structured unlabeled samples. As the number of labeled samples is substantially smaller than the number of structured unlabeled samples, λ should be smaller than 1 to balance the number of structured unlabeled samples. Experiments were performed to determine the proper value of λ for an FNN with one hidden layer of 100 nodes, and the ReLu activation function was used. As shown in Fig. 7, we can see that the prediction errors of the labeled samples increase as λ increases because the regression losses and the prediction errors of the labeled samples are highly correlated. Adding less weight to the regression losses will surely degrade the performance of the network on labeled samples.

The prediction errors of the structured unlabeled samples and test samples initially decrease and then increase with increasing λ. When λ is too small, the networks overfit the labeled samples, and when λ is too large, the decreasing structure losses drive the learning in the wrong direction. The best value for λ is approximately $0.01$ for the FNNs used in these experiments.

*3) Relation Between the Prediction Errors and Number of Labeled Samples:* Experiments were performed to study the relation between the prediction errors and the number of labeled samples. Labeled samples were randomly generated, and different numbers of labeled samples were used in these experiments. The structured unlabeled samples used in the training were the same as in the experiments above. An FNN with one hidden layer of 100 nodes was trained, and the ReLu activation function was used. The experimental results are shown in Fig. 8.

Intuitively, the prediction errors should decrease with increasing number of labeled samples, and the prediction errors do decrease with increasing number of labeled samples. However, the downtrend of the prediction errors becomes increasingly flat. The reason is the distribution of the labeled samples used in the training. To this end, we studied the relation between the prediction errors and the distributions of the labeled samples below.

*4) Relation Between the Prediction Errors and the Distributions of the Labeled Samples:* As mentioned above, the distribution of the labeled samples in the latent space may influence the prediction errors. To this end, experiments were performed to

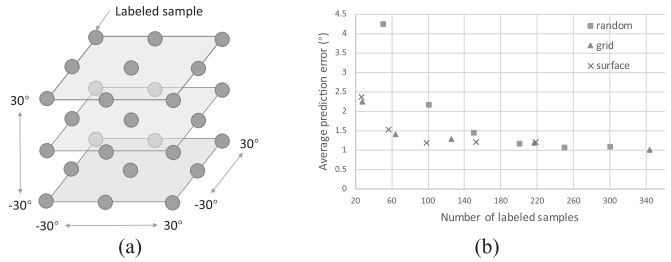



Fig. 9. Relation between the prediction errors and the distributions of the labeled samples. (a) Distribution of the labeled samples [triangle marks in (b)] in the latent space. (b) Relation between the prediction errors of the test samples and the distributions of the labeled samples. The square marks represent the results of randomly distributed labeled samples, the triangle marks represent the results of labeled samples distributed as in the cube grids in (a), and cross marks represent labeled samples distributed on the surfaces of the cube in (a).

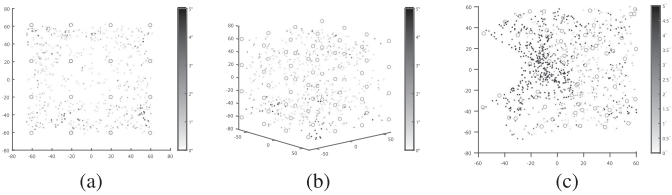



Fig. 10. Error distributions of the test samples, where the gray dots represent the predicted positions of the test samples in the latent space the red circles represent the predicted positions of the labeled samples; the structured unlabeled samples are not displayed. The values of the prediction errors are shown using the color bar. (a) Error distribution of an FNN trained with 64 labeled samples distributed on the nodes of the cubic grids. (b) Same error distribution of the test samples as (a) from another viewpoint. (c) Error distribution of an FNN trained with 64 randomly distributed labeled samples.

study the relation between the prediction errors and the distributions of the labeled samples. Three types of distributions of the labeled samples were studied, and different numbers of labeled samples were considered. An FNN with one hidden layer of 100 nodes was trained, and the ReLu activation function was used. The structured unlabeled samples used in the training were the same as in the experiments above.

As shown in Fig. 9(b), the Y-axis represents the average prediction error of the test samples. The square marks represent experiments using randomly distributed labeled samples, which are the same as in the experiments in Fig. 8. To make the distributions of the labeled samples more uniform, the 3D latent space was divided into grids, as shown in Fig. 9(a). In addition, the labeled samples distributed on the nodes of the grids were used in the experiments; the results are indicated with triangle marks. To further reduce the number of labeled samples, only the nodes on the outer surface of the cube grid were used in the other set of experiments, and the results are denoted as cross marks.

We can see from the results that using labeled samples with well-designed distributions can achieve higher prediction accuracies on test samples than can random distributions. Using labeled samples distributed on the outer surface achieves similar performance to labeled samples distributed on the nodes of the cubic grid. The reason can be seen in Fig. 10. Fig. 10(a) and (b) show the error distributions of the test samples of a trained FNN, where the positions of the spots are the predicted positions in the latent space (the latent representation) and darker spots represent higher prediction errors. The FNN was trained with 64 labeled samples distributed on the nodes of a cubic grid. Fig. 10(c) shows the error distribution of an FNN trained with 64 randomly distributed labeled samples. The same structured unlabeled samples were used in the training of these two FNNs. We can see that the error distributions in Fig. 10(a) and (b) are even. In addition, the error distribution in Fig. 10(c) is uneven. Samples with large errors accumulate on the left-hand side of the figure. This is because these areas lack labeled samples. Without labeled samples, the framework of the latent space is not built, and thus, the prediction errors in these area are not reduced.

Fig. 11. Sample images from EPFL.

### C. Experiments

We evaluate the proposed methods using the EPFL Multi-view Car Dataset [36]. The dataset has continuous orientation annotations available, which can be regarded as latent representations.

*1) EPFL Dataset:* The EPFL dataset contains 20 sequences of images of vehicles captured at a car show where each sequence contains images of the same instance of the vehicle captured with various orientations. Each image is given a ground truth orientation. We choose this dataset to test our method because the cars contain city cars, sedans, Sport Utility Vehicle, and concept cars, as shown in Fig. 11.

To compare our method with state-of-the-art methods, we also use the first 10 sequences of vehicles (1179 images) for training and evaluation, the second 10 sequences of vehicles (1120 images) for testing. Ground truth bounding boxes that come with the dataset are used to crop out the image regions, and then the image regions are resized to $100 \times 100$ pixel images.

The training set and the testing set contain completely different types of cars with different colors, shapes, backgrounds, and lighting conditions. Directly using original RGB (red, green, blue) images to train the network will lead to serious overfitting. To reduce overfitting, the RGB image is decomposed to a gray image, edge image, and red channel image. For the edge image, we use the Scharr transform [37] to obtain the edge map. For the red channel image, we first transform the RGB image to an HSV (hue, saturation, value) image and then extract the red region by thresholds. We choose the red channel because the taillights are red, and the red channel images can reduce "flipping" errors

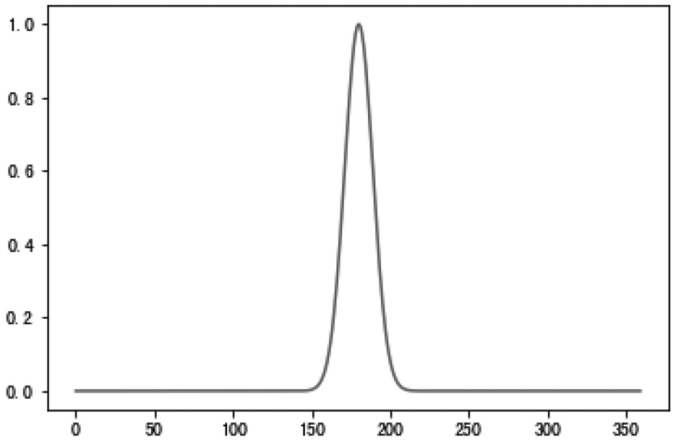



Fig. 12. Distribution representing the 180°.

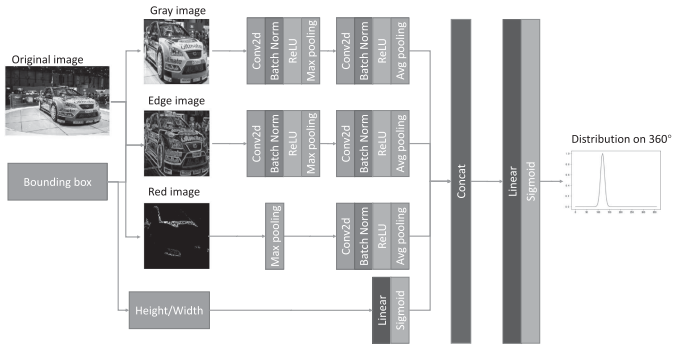



Fig. 13. Network architecture of the proposed method.

according to our experiments. Data augmentation is also used to reduce overfitting. Specifically, the bounding box is enlarged left, right, top, and bottom by 5 pixels to generate four more images for each image. When predicting the orientations on the testing set, we average the five predictions of the original image and the four augmented images as the final prediction.

*2) CNN Architecture:* The EPFL dataset contains 20 different kinds of vehicles. Predicting the orientations of cars in this dataset is a difficult task. CNN architectures have been proven in the literature to have the ability to solve complex tasks. As mentioned above, the proposed method can also be embedded into CNN architectures; therefore, we designed a CNN architecture shown in Fig. 13 to solve the problem.

The gray image, edge image, and red channel image are fed into three separate CNN blocks. In addition, the ratio of height to width of the original image before resizing is fed into a feed-forward block. The outputs of the four blocks are concatenated, and then fed into a feed-forward block.

According to [38], discretization-based approaches significantly outperform regression on continuous angles with CNN architectures, and discretization alone will confuse the training algorithm since there are small differences in appearances among neighboring orientations. Inspired by their work, we transform the angles to distributions on 360°. For example, the distribution of 180° is

$$e^{-\frac{(x-\alpha)^2}{\sigma^2}} \tag{6}$$

where $x = 0, 1, ..., 359$ is the index of the dimensions of the distribution, $\alpha$ is the angle before transformation, $\sigma$ is a manually selected coefficient used to adjust the distribution; in this paper, we select $\sigma = \sqrt{160}$. The distribution for $\alpha = 180°$ is shown in Fig. 12. The outputs of the CNN architecture are thus a 360-dimensional vector. The objective of the training is to reduce the MSE between the outputs of the CNN and the ground truth distributions. The predicted angles can be retrieved by finding the highest peak of the distributions.

The advantage of transforming angles to distributions is that it will not confuse the training algorithm, because the activations of the neurons in the output layer are reduced gradually with the distances to the ground truth angle, instead of discrete values of 1 and 0 s.

*3) Training Details and Results:* The experiment was conducted using the PyTorch deep learning framework on a NVIDIA 1080Ti GPU with 12 GB memory. The parameters of the CNN are trained by the Adam algorithm [39] with an initial learning rate of 0.001. We split the first 10 sequences of vehicles for training and evaluation, among which 100 images are randomly selected for evaluation and the remaining images are used for training. We evaluate the proposed network with three different combinations of labeled samples and structured unlabeled samples. For our approach 1, 1079 labeled samples in the first 10 sequences are used for training and the remaining 100 labeled samples are used for evaluation. For our approach 2, we take one out of every ten adjacent frames in the first 10 sequences as the labeled samples, and the number of labeled samples is 123. We randomly take subsequences with a length of 10–20 from the first 10 sequences as the structured unlabeled samples, because there are repeated samples, the number of unlabeled samples is 1389. For our approach 3, only the 123 labeled samples are used without the unlabeled samples. Our approach 3 is used to verify the importance of using structured unlabeled samples with

TABLE II
COMPARISON WITH THE EXISTING METHODS ON THE EPFL DATASET

| Methods | MeanAE | MedianAE | Labeled | Unlabeled |
|---|---|---|---|---|
| Our approach 1 | **9.28** | 3.5 | 1079 | 0 |
| Our approach 2 | 12.02 | 3.65 | **123** | 1389 |
| Our approach 3 | 17.22 | 4.78 | **123** | 0 |
| Fenzi *et al.* (2015) [40] | 13.6 | **3.3** | 1179 | 0 |
| He *et al.* (2014) [41] | 15.8 | 6.2 | 1179 | 0 |
| Yang *et al.* (2017) [42] | 20.30 | 3.36 | 1179 | 0 |
| Fenzi *et al.* (2014) [43] | 23.28 | N/A | 1179 | 0 |
| Hara *et al.* (2017) [44] | 23.81 | N/A | 1179 | 0 |
| Zhang *et al.* (2013) [45] | 24.00 | N/A | 1179 | 0 |
| Yang *et al.* (2014) [46] | 24.1 | 3.3 | 1179 | 0 |
| Hara *et al.* (2014) [47] | 24.24 | N/A | 1179 | 0 |
| Fenzi *et al.* (2013) [48] | 31.27 | N/A | 1179 | 0 |
| Torki *et al.* (2011) [49] | 33.98 | 11.3 | 1179 | 0 |
| Teney *et al.* (2014) [50] | 34.7 | 5.2 | 1179 | 0 |
| Redondo *et al.* (2014) [51] | 39.8 | 7 | 1179 | 0 |
| Ozuysal *et al.* (2009) [36] | 46.5 | N/A | 1179 | 0 |

The performance is verified in MeanAE and MedianAE.

Fig. 14. Representative results obtained by the proposed method (our approach 2, labeled = 123, unlabeled = 1389). A ground truth orientation (yellow) and the predicted orientation (red) are indicated in a circle. Each row contains 10 example results from a testing sequence. From left to right, images with seven frames apart are selected, starting from the first frame.

structure loss. All three approaches use all the frames in the second 10 sequences of the EPFL dataset for testing.

Training stops when the regression loss no longer decrease anymore on the evaluation dataset for more than 2 epochs. The performance of the algorithm is verified by mean absolute error (MeanAE) and median absolute error (MedianAE) in degree following the practice in the literature. The MedianAE evaluates the performance of the method after removing very large errors, also known as "flipping" errors.

In Table II, we present the results from the literature and the results of our three approaches with the same network architecture (our approach 1, our approach 2, and our approach 3). As seen, our approach 1 obtains a relative improvement in the MeanAE of approximately 31.8% with respect to state of the art [40]. Our approach 2 uses only 123 labeled samples with the regression loss and 1389 structured unlabeled samples with the structure loss but still outperforms the state of the art [40] by 11.6% in MeanAE. In addition, our approach 3 using only the 123 labeled samples with the regression loss is not as good as our approach 2, which can prove the effectiveness of the structure loss.

The information of the numbers on the labeled samples and the structured unlabeled samples used for training are included in the table (labeled, unlabeled). Methods in the literature use all the frames in the first 10 sequences in the EPFL dataset (1179 frames) and do not use any unlabeled samples.

Finally, we show the representative results in Fig. 14 with ground truth bounding boxes overlaid on the images, and a ground truth orientation and the predicted orientation indicated in a circle.

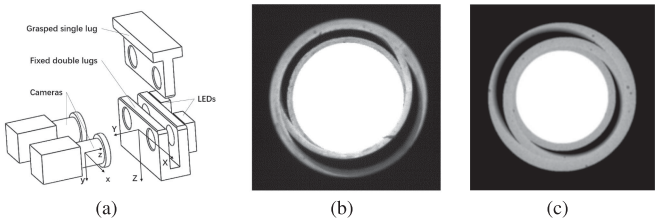



Fig. 15. Pin-in-hole assembly system. (a) Single-lug must be inserted into the double-lugs, and the holes on the single-lug and the double-lugs must be aligned. Cameras are used in the pin-in-hole assembly system to capture images of the holes. (b) Real image captured by the camera when the single-lug is inserted into the double-lugs. (c) Example of simulated images used for evaluation, both shadows and image defects are simulated, except for the light reflections.

TABLE III
COMPARISON WITH BASELINE ON THE COAXIAL ERROR DETECTION DATASET

| Methods | MD (pixel) | Labeled | Unlabeled |
|---|---|---|---|
| Baseline | 4.2 | 25 | 0 |
| **Ours** | **2.5** | 25 | 689 |

Performance is measured in mean distance (MD) between the predicted center and the ground truth center. The number of labeled samples and structured unlabeled samples used in the training is also given.

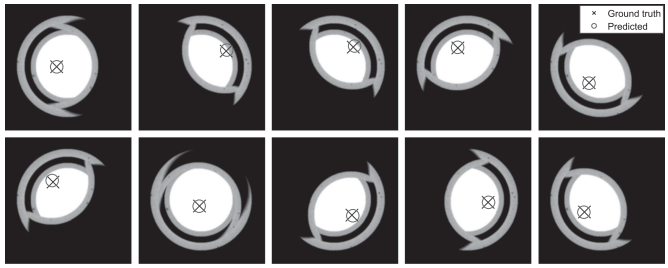



Fig. 16. Representative results are obtained by the proposed method. A ground truth center of the hole on the single-lug (blue cross) and the predicted center (red circle) are indicated in the image.


Fig. 17. Two applications. (a) Robotic system in visual servoing tasks. (b) Image of the flange plate used in the tasks.

### D. Applications

Our method (FNN + PCA + full objective) is evaluated in three challenging applications: coaxial error detection, visual servoing, and indoor location.

*1) Coaxial Error Detection:* The proposed method was originally designed to solve the coaxial error detection problem in pin-in-hole assembly tasks [52]. In automated pin-in-hole assembly tasks, the coaxial error of the holes on different lugs must be measured in a timely manner. Manual measurement cannot meet the real-time requirement; a possible solution is to train a neural network to measure the coaxial errors directly from images. Traditional training methods require a large number of labeled samples; however, measuring coaxial error is a time-consuming task, and we need to reduce the demands of labeled samples. Therefore, the proposed method is designed to solve the problem.

The pin-in-hole assembly system is shown in Fig. 15(a). A representative image captured by the camera is shown in Fig. 15(b). We construct a dataset of simulated images to evaluate the proposed method. An example of the simulated images is shown in Fig. 15(c). Both shadows and image defects are simulated, except for the light reflections. The objective of the deep neural network is to predict the position of the center of the hole on the single lug in the image. The position of the center can be regarded as a 2D latent representation of the image. The results are shown in Table III. Our method shares the same network architecture with the baseline model, except that structure loss using structured unlabeled samples is added. With the structure loss, the detection errors are reduced by 40.5%. Representative results obtained by our method are shown in Fig. 16.

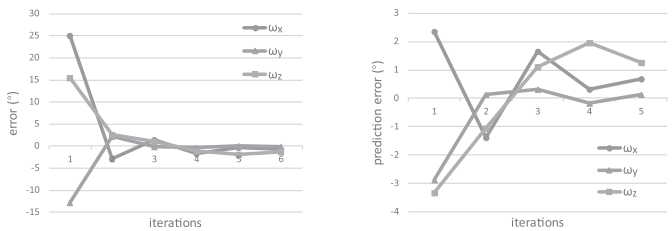
*2) Visual Servoing:* For visual servoing, we performed our experiments on the robotic system shown in Fig. 17(a). Only rotations were considered in these experiments. Fig. 17(b) is the image of a flange plate captured by the camera. The flange plate is the target used in the visual servoing. We regarded the pose of the flange plate in the image as the target pose, where the Eulerian angle $\left[\omega_{\mathrm{x}}\ \omega_{\mathrm{y}}\ \omega_{\mathrm{z}}\right]^{\top} = \left[0\ 0\ 0\right]^{\top}$. A total of 125 labeled samples and 1673 structured unlabeled samples were used to train an FNN with one hidden layer of 100 neurons. The ReLu activation function was used in the network, and PCA was performed for down-sampling. The original images are $54 \times 54$ pixels, and the image vectors after PCA down-sampling are 41 dimensions column vectors. A total of 100 randomly distributed samples were used to test the prediction errors of the network, and the average prediction error after training for 100 epochs is $4.67^{\circ}$.

Visual servoing tasks were performed using the trained network. Specifically, the pose of the flange plate relative to the target pose is directly predicted from each captured image. In addition, rotations are performed to make the flange plate reach the target pose. Table IV shows the initial and final positioning errors of 20 experiments. Although the average prediction error of the trained network is $4.67^{\circ}$, the final positioning error is less than $2^{\circ}$ because the final positioning error is determined by the prediction errors near the target pose. Fig. 18 shows the graphs plotted for one of the experiments. We can see that the proposed method achieves quick convergence.

*3) Indoor Localization:* We adapted our method to the task of indoor localization. In this task, we want to estimate the location of the camera based on the captured images. The simulation scene is shown in Fig. 19. In addition, training samples were gathered by capturing images using a virtual camera. The virtual

TABLE IV
THE EXPERIMENTAL RESULTS OF VISUAL SERVOING TASKS

| **Initial Pose Error (°)** | | | **Final Pose Error (°)** | | |
|---|---|---|---|---|---|
| $\Delta\omega_x$ | $\Delta\omega_y$ | $\Delta\omega_z$ | $\Delta\omega_x$ | $\Delta\omega_y$ | $\Delta\omega_z$ |
| -6.47 | 9.33 | -19.73 | -0.54 | 0.02 | -1.36 |
| 12.36 | -28.09 | -13.38 | -0.59 | -0.09 | -1.30 |
| -27.23 | -24.17 | 19.41 | -0.37 | 0.11 | -1.62 |
| 11.69 | -10.97 | 27.01 | -0.62 | -0.07 | -1.40 |
| -27.93 | -3.68 | -7.11 | -0.70 | -0.11 | -1.35 |
| 15.93 | 17.71 | -18.79 | -0.67 | -0.02 | -1.48 |
| -0.61 | -3.26 | 8.78 | -0.64 | -0.13 | -1.32 |
| 12.56 | 15.28 | -13.44 | -0.58 | -0.03 | -1.60 |
| 10.78 | 9.31 | -20.24 | -0.60 | -0.02 | -1.46 |
| -22.86 | -0.10 | 27.58 | -0.44 | -0.02 | -1.83 |
| -9.58 | 5.12 | -16.57 | -0.63 | -0.13 | -1.29 |
| 15.08 | -14.69 | 0.36 | -0.63 | -0.05 | -1.48 |
| 11.94 | 23.45 | 27.56 | -0.66 | -0.06 | -1.40 |
| 2.83 | -21.68 | -21.04 | -0.71 | -0.11 | -1.40 |
| -14.55 | 20.44 | -14.74 | -0.51 | 0.02 | -1.72 |
| 18.86 | -15.39 | 25.76 | -0.75 | -0.12 | -1.36 |
| -9.00 | -18.20 | -14.93 | -1.13 | -0.09 | -1.63 |
| 6.96 | -1.60 | -8.90 | -0.74 | -0.09 | -1.39 |
| 19.85 | 5.12 | 2.98 | -0.76 | -0.04 | -1.57 |
| 25.03 | -12.85 | 15.43 | -0.68 | -0.14 | -1.26 |




(a) (b)

Fig. 18. Self-supervised-learning-based visual servoing on a flange plate. (a) Rotational errors. (b) Prediction errors of the current pose relative to the target pose.

Fig. 19. Indoor scene used for indoor localization task.

camera can move in a rectangular area of 4 m $\times$ 2 m. The height and orientation of the virtual camera are fixed.

A total of 231 labeled samples, 1864 structured unlabeled samples, and 100 test samples were gathered. An FNN with one hidden layer of 100 nodes was trained. The ReLu activation function was used in the FNN, and PCA was performed on the samples before training. $54 \times 54$ pixels original images are down-sampled by PCA to 35 dimensions column vectors. The

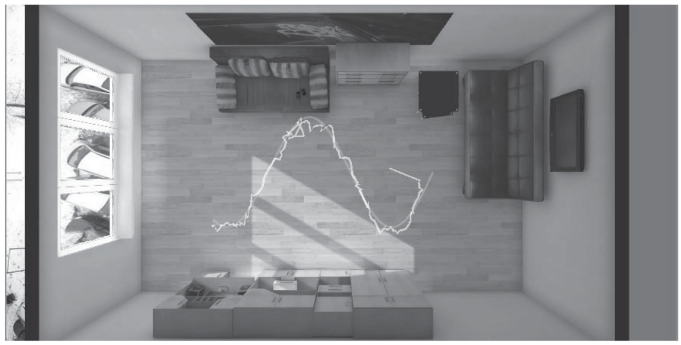


Fig. 20. Real trajectory (red) and the predicted trajectory (green) by the trained FNN.

average prediction error after training is $0.1m$. The trained FNN was further used to predict the trajectory of the virtual camera. The results are shown in Fig. 20. The red trajectory is the real trajectory along which the virtual camera moves. Images were captured along the trajectory, and then, the positions of the virtual camera were predicted by the trained FNN using the captured images as the inputs. The green trajectory in Fig. 20 is the predicted trajectory. The predicted trajectory is in good agreement with the real trajectory.

## V. CONCLUSION

We have presented a specified latent representation method from raw images using neural networks with only a few labeled samples. These labeled samples are used to set up the framework of the latent space using the regression loss with supervised learning; meanwhile, these labeled samples are mapped to the framework nodes in the latent space. However, the prediction accuracy of the shape between framework nodes is low. Then, a large number of structured unlabeled samples are used to shape the free space between the framework nodes using the structure loss with the proposed self-supervised learning technique. Different network architectures embedded with our proposed method are studied. Car orientation detection, coaxial error detection, visual servoing, and indoor localization are used to demonstrate the generality of our proposed method. The results show that our proposed self-supervised learning method possesses the advantages of supervised learning and unsupervised learning for a specified latent representation.

The limitation of the proposed method is that the unlabeled samples we use must have specific structure information; however, most available datasets on the Internet are not organized in structures [53]. This limitation could be mitigated by generating structured unlabeled samples, for example, extracting structure information from videos by correlating adjacent frames [54]–[56]. Our future work will focus on solving data shortage with structure information by extracting structure information in videos and constructing dataset. Besides, frame sequences can be regarded as time series, Hidden Markov Model (HMM) could be applied to characterize the frame sequences, where the latent representations are the hidden states in HMM [57], [58]. We will explore learning methods for latent representations based on HMM in our future work.

## REFERENCES


[1] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2746–2754.

[2] A. Byravan, F. Leeb, F. Meier, and D. Fox, "SE3-Pose-Nets: Structured deep dynamics models for visuomotor planning and control," 2017, arXiv: 1710.00489.

[3] R. Jonschkowski, R. Hafner, J. Scholz, and M. Riedmiller, "PVEs: Position-velocity encoders for unsupervised learning of structured state representations," 2017, arXiv: 1705.09805.

[4] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016, pp. 512–519.

[5] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," Colorado Univ. Boulder Dept. Comput. Sci., *Tech. Rep.*, 1986.

[6] T. Schmah, G. E. Hinton, S. L. Small, S. Strother, and R. S. Zemel, "Generative versus discriminative training of RMBs for classification of fMRI images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1409–1416.

[7] G. W. Taylor and G. E. Hinton, "Factored conditional restricted Boltzmann machines for modeling motion style," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*. New York, NY, USA: ACM, 2009, pp. 1025–1032.

[8] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biol. Cybern.*, vol. 59, no. 4–5, pp. 291–294, Sep. 1988.

[9] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and Helmholtz free energy," in *Proc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 3–10.

[10] Y. Bengio and O. Delalleau, "Justifying and generalizing contrastive divergence," *Neural Comput.*, vol. 21, no. 6, pp. 1601–1621, 2009.

[11] Q. Bateux, E. Marchand, J. Leitner, and F. Chaumette, "Visual servoing from deep neural networks," 2017, arXiv: 1705.08940.

[12] G. Dahl *et al.*, "Phone recognition with the mean-covariance restricted Boltzmann machine," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, vol. 1. USA: Curran Associates Inc., 2010, pp. 469–477.

[13] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-R. Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010.

[14] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. 12th Annu. Conf. Int. Speech Commun. Assoc.*, 2011.

[15] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 14–22, Jan. 2012.

[16] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[17] G. Hinton *et al.*,"Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[19] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Providence, RI, USA, 2012.

[20] S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio, and X. Muller, "The manifold tangent classifier," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2294–2302.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[22] G. E. Hinton, "Learning distributed representations of concepts," in *Proc. 8th Annu. Conf. Cognit. Sci. Soc.*, vol. 1. Amherst, MA, 1986, p. 12.

[23] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[24] Y. Bengio, "Neural net language models," *Scholarpedia*, vol. 3, no. 1, p. 3881, 2008.

[25] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[26] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proc. ICML Workshop Unsupervised Transfer Learn.*, 2012, pp. 17–36.

[27] G. Mesnil *et al.*, "Unsupervised and transfer learning challenge: A deep learning approach," in *Proc. Int. Conf. Unsupervised Transfer Learn. Workshop*, vol. 27. JMLR.org, 2011, pp. 97–111.

[28] I. J. Goodfellow, A. Courville, and Y. Bengio, "Spike-and-slab sparse coding for unsupervised feature discovery," 2012, arXiv:1201.3382.

[29] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proc. 28th Int. Conf. Mach. Learn. (ICML-11)*, 2011, pp. 513–520.

[30] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[31] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, and K. R. Mller, Eds. Berlin: Springer, 2012, pp. 639–655.

[32] Y. Bengio, H. Larochelle, and P. Vincent, "Non-local manifold Parzen windows," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 115–122.

[33] Y. Bengio and Y. Lecun, "Scaling learning algorithms towards AI," in *Large-scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. Cambridge, MA, USA: MIT Press, 2007, pp. 1–41.

[34] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, no. 1–3, pp. 37–52, 1987.

[35] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *IEEE Potentials*, vol. 13, no. 4, pp. 27–31, Oct-Nov. 1994.

[36] M. Ozuysal, V. Lepetit, and P. Fua, "Pose estimation for category specific multiview object localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009.

[37] D. Kroon, "Numerical optimization of kernel based image derivatives," *Short Paper Univ. Twente*, 2009.

[38] K. Hara, R. Vemulapalli, and R. Chellappa, "Designing deep convolutional neural networks for continuous object orientation estimation," 2017, arXiv: 1702.01499.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, arXiv:1412.6980.

[40] M. Fenzi, L. Leal-Taix é, J. Ostermann, and T. Tuytelaars, "Continuous pose estimation with a spatial ensemble of fisher regressors," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, Dec. 2015.

[41] K. He, L. Sigal, and S. Sclaroff, "Parameterizing object detectors in the continuous pose space," in *Proc. Eur. Conf. Comput. Vis.*, Springer, Cham, Switzerland, 2014, pp. 450–465.

[42] D. Yang, Y. Qian, K. Chen, E. Berki, and J. K. Kamarainen, "Hierarchical sliding slice regression for vehicle viewing angle estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 2035–2042, Jun. 2018.

[43] M. Fenzi and J. Ostermann, "Embedding geometry in generative models for pose estimation of object categories," *Proc. British Mach. Vis. Conf.*, vol. 1, no. 2, p. 3, 2014.

[44] K. Hara and R. Chellappa, "Growing regression tree forests by classification for continuous object pose estimation," *Int. J. Comput. Vis.*, vol. 122, no. 2, pp. 292–312, 2017.

[45] H. Zhang, T. El-Gaaly, A. M. Elgammal, and Z. Jiang, "Joint object and pose recognition using homeomorphic manifold analysis," in *Proc. AAAI Conf. Artif. Intell.*, Bellevue, WA, USA, vol. 2. 2013, p. 5.

[46] L. Yang, J. Liu, and X. Tang, "Object detection and viewpoint estimation with auto-masking neural network," in *Proc. Eur. Conf. Comput. Vis.* Springer, Cham, Switzerland, 2014, pp. 441–455.

[47] K. Hara and R. Chellappa, "Growing regression forests by classification: Applications to object pose estimation," in *Proc. Eur. Conf. Comput. Vis.*. Springer, Cham, Switzerland, 2014, pp. 552–567.

[48] M. Fenzi, L. Leal-Taixé, B. Rosenhahn, and J. Ostermann, "Class generative models based on feature regression for pose estimation of object categories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, Jun. 2013.

[49] M. Torki and A. Elgammal, "Regression from local features for viewpoint and pose estimation," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Barcelona, Spain, Nov. 2011.

[50] D. Teney and J. Piater, "Multiview feature distributions for object detection and continuous pose estimation," *Comput. . Image Understanding*, vol. 125, pp. 265–282, 2014.

[51] C. Redondo-Cabrera, R. López-Sastre, and T. Tuytelaars, "All together now: Simultaneous object detection and continuous pose estimation using a hough forest with probabilistic locally enhanced voting," in *Proc. British Mach. Vis. Conf.*, 2014, pp. 1–12.

[52] C. Liu, J. Xu, J. Zhao, H. Chen, N. Xi, and K. Chen, "Non-vector space visual servoing for multiple pin-in-hole assembly by robot," in *Proc. IEEE Workshop Adv. Robot. Social Impacts (ARSO)*, Shanghai, China, Jul. 2016.
[53] A. Kuznetsova *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," 2018, arXiv: 1811.00982.
[54] Y. Pan, Y. Li, T. Yao, T. Mei, H. Li, and Y. Rui, "Learning deep intrinsic video representation by exploring temporal coherence and graph structure," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 3832–3838.
[55] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for correlation filter based tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, Jul. 2017.
[56] R. Gao, D. Jayaraman, and K. Grauman, "Object-centric representation learning from unlabeled videos," in *Proc. Asian Conf. Comput. Vis.*. Springer, Cham, Switzerland, 2016, pp. 248–263.
[57] W. Zucchini, I. L. MacDonald, and R. Langrock, in *Hidden Markov Models for Time Series: An Introduction Using R*. Boca Raton, FL, USA: Chapman and Hall/CRC, 2016.
[58] E. Savku and G.-W. Weber, "A stochastic maximum principle for a Markov regime-switching jump-diffusion model with delay and an application to finance," *J. Optim. Theory Appl.*, vol. 179, no. 2, pp. 696–721, Nov. 2018.

**Chicheng Liu** received his B.E. degree in mechanical engineering in 2013 from Tsinghua Unviersity, Beijing, China, where he is currently pursuing his Ph.D. in the Department of Mechanical Engineering.

His research interests include visual servoing, robotic systems, and machine intelligence.

**Libin Song** received B.E. and Ph.D. degrees in mechanical engineering from Tsinghua University, Beijing, China, in 2000 and 2008, respectively.

He is currently an Assistant Professor in the Department of Mechanical Engineering, Tsinghua University, Beijing, China.

His research interests include advanced manufacturing, special robot, and biomedical instrument.

**Jiwen Zhang** received his M.S. and Ph.D. degrees in mechanical engineering from Tsinghua University, Beijing, China, in 2008 and 2013, respectively.

From 2017 to the present, he has been a Research Assistant with the Department of Mechanical Engineering at Tsinghua University, where he has been working on motion planning for legged robots. He is the author of over 15 technical publications, proceedings, editorials, and books.

His research interests include mobile robotics, motion planning, and automatic control systems. He is also an active member of the RoboCup Societies.

**Ken Chen** received his Ph.D. in mechanical engineering from Zhejiang University, Hangzhou, China, in 1987.

He is currently a Professor in the Department of Mechanical Engineering, Tsinghua University, Beijing, China.

His research interests include robotics and intelligent control, humanoid robots, microrobots and small robots, medical and space robots, manufacturing automation systems, and hydraulic servo systems.

**Jing Xu** received his Ph.D. in mechanical engineering from Tsinghua University, Beijing, China, in 2008. He was a Postdoctoral Researcher in the Department of Electrical and Computer Engineering, Michigan State University, East Lansing.

He is currently an Associate Professor in the Department of Mechanical Engineering, Tsinghua university, Beijing, China.

His research interests include vision-guided manufacturing, image processing, and intelligent robotics.