IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, USA, October 2020.

Category-Level 3D Non-Rigid Registration from Single-View RGB Images

Diego Rodriguez*, Florian Huber*, and Sven Behnke

***Abstract*— In this paper, we propose a novel approach to solve the 3D non-rigid registration problem from RGB images using Convolutional Neural Networks (CNNs). Our objective is to find a deformation field (typically used for transferring knowledge between instances, e.g., grasping skills) that warps a given 3D canonical model into a novel instance observed by a single-view RGB image. This is done by training a CNN that infers a deformation field for the visible parts of the canonical model and by employing a learned shape (latent) space for inferring the deformations of the occluded parts. As result of the registration, the observed model is reconstructed. Because our method does not need depth information, it can register objects that are typically hard to perceive with RGB-D sensors, e.g. with transparent or shiny surfaces. Even without depth data, our approach outperforms the Coherent Point Drift (CPD) registration method for the evaluated object categories.**

I. Introduction

Registering 3D objects in a non-rigid way is essential for many real world applications, including robot manipulation [1], [2], human body analysis [3], [4] and grasp planning [5], [6]. This registration plays a key role for transferring domain knowledge, such as the transfer of control poses for approaching and grasping objects. This knowledge transfer is inspired by the observation that objects with similar shape and usage can be manipulated in an analogous manner—adapting knowledge from previous successful experiences to novel observed instances. Following this idea, in this work, we register non-rigidly a canonical model towards novel instances of the same object category.

The 3D non-rigid registration problem is often addressed based on RGB-D images or 3D scans [7]–[9]. However, several objects cannot be measured well by depth sensors, e.g., transparent drinking bottles. By registering objects from RGB images, we attempt to address this issue.

The registration from a single RGB image is a challenging problem for several reasons. First, 3D deformations need to be calculated without depth information, i.e., a mapping between 2D colored pixels and 3D deformations has to be established. Second, from a single-view RGB image the object is not fully observable and occluded parts have to be reconstructed. Third, this reconstruction problem is ambiguous; several plausible shapes can explain a single observation.

* Authors with equal contribution. All authors are with the Autonomous Intelligent Systems (AIS) Group, Computer Science Institute VI, University of Bonn, Germany rodriguez@ais.uni-bonn.de

This work was funded by grant BE 2556/16-1 (Research Unit FOR 2535 Anticipating Human Behavior) of the German Research Foundation (DFG) and an Amazon Research Award.

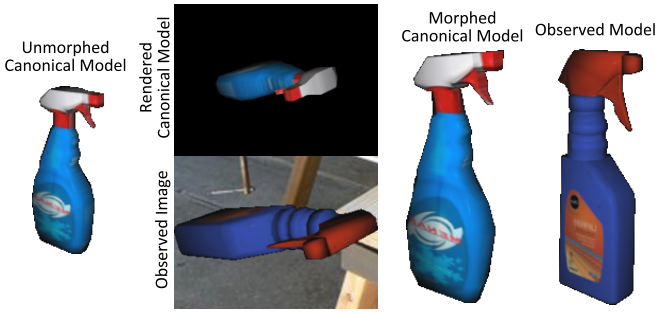

Fig. 1. A canonical model of a category is deformed based on a single-view RGB image. The model is reconstructed without any depth information from partially observing the object. Models are shown at the same scale to show the large deformation the canonical model has undergone.

In this paper, we propose a novel approach that is able to infer 3D non-rigid deformations from single-view RGB-only images of objects belonging to the same category[1] (Fig. 1). This is done by leveraging on realistic 3D object models. A deep Convolutional Neural Network (CNN) is trained to infer deformations in the x, y and z axes, given the observed image of an object and a rendered image of the canonical model. The training images are generated by rendering 3D realistic models and calculating ground truth deformations using the Coherent Point Drift (CPD) method. By using a learned shape space that describes the typical geometrical variability of the object category, our approach is able to reconstruct the observed instances and provides deformations even for the non-observable parts. Thus, the reconstructed object exhibits a category-like shape. The resulting deformation field allows to transfer category-level knowledge, e.g. grasping skills, to novel instances without further training.

The core contribution of this paper is to introduce a novel approach that registers non-rigidly a 3D model towards an object observed by a single RGB image. In addition, we develop a method to generate a dataset suitable for non-rigid registration tasks. This method is able to synthetically augment the number of instances of a category to overcome the scarcity of high-quality 3D textured models.

II. Related Work

A. Rendering for Deep Learning

The use of synthetic data to generate training samples has been widely adopted in the deep learning community for object detection, semantic segmentation and pose estimation tasks [10]–[13]. One of the first successful attempts was proposed by Tobin *et al.* [14], who trained an object detection

[1] Video: www.ais.uni-bonn.de/videos/IROS_2020_Rodriguez

network using only synthetic data and were able to transfer the network to real-world applications. More recently, Tremblay *et al.* [15] achieved state-of-the-art performance on 6-DoF pose estimation by using only synthetic data. Our work is partially inspired by pose refinement approaches [16], that demonstrated that the rigid registration problem can be solved by the use of rendered images.

B. Non-Rigid Registration

Most of the approaches that address the 3D non-rigid registration problem assume the availability of full volumetric data. These approaches mainly differ by the constraints imposed on how the points should move to match the observation including: thin splines [4], [7], conformal maps [17], [18], and coherent motion [19]. However, these approaches do not perform well with partially observed objects. In our previous work [20], we incorporated a shape space of similar instances to address this issue. Plausible geometries are inferred and a deformation field is provided for the non-observable parts, but depth information is required. In contrast, the method presented in this paper registers a canonical model to novel object instances in a non-rigid manner without depth information—only a single-view RGB image of the observed object is required. This is possible thanks to category-level prior knowledge of the objects represented in a CNN.

C. Shape Reconstruction

For 3D shape completion, several approaches have been proposed, including radial basis functions [21], surface primitives [22], and Laplacian mesh optimization [23]. Data-driven approaches have been recently presented for 3D completion tasks. Choy *et al.* [24] learn a mapping from object images to 3D shapes from a large collection of synthetic data by means of a Recurrent Neural Network (RNN). In [25], a 3D model is completed given a single depth image by transferring symmetries and surfaces from an exemplary database. By integrating deep generative models with adversarially learned shape priors, Wu *et al.* [26] are able to complete and reconstruct 3D shapes from single view images.

Similar to our work, the approaches proposed by Wu *et al.* [26] and Choy *et al.* [24] also attempt to reconstruct a 3D model based on a RGB image. While both approaches achieve good results when estimating shapes, they do not provide any estimate of the deformations between instances or any kind of registration. Our approach, on the other hand, reconstructs the observed object and provides deformation fields enabling knowledge transfer to novel instances directly without the need for costly additional registrations posterior to the reconstruction.

III. Background

In our approach, we infer a deformation field that warps a canonical model into an observed instance from a single-view RGB image. We train a deep CNN in a supervised manner to infer a deformation field from the observed input image. This field is represented as three output feature maps representing the deformations in the x, y and z axes of the object. The target data for training is calculated using the Coherent Point Drift [19] which is briefly introduced below. To infer the deformation field of the non-observable parts, a shape (latent) space of the object category is constructed (Sec. III-B) [9], [20].

A. Coherent Point Drift

For two sets of D-dimensional points, $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$ and $\mathbf{Y} = (\mathbf{y}_1, ..., \mathbf{y}_M)^T$, CPD outputs a deformation field mapping $\mathbf{Y}$ towards $\mathbf{X}$. For this purpose, the points of $\mathbf{Y}$ are considered as centroids of a Gaussian Mixture Model (GMM) and the points of $\mathbf{X}$ are considered as samples drawn from these GMMs. Equal isotropic covariances σ^2 and equal membership probabilities $P(m) = \frac{1}{M}$ are used for all GMM components. CPD outputs a deformed point set $\mathcal{T}$ as result of the maximization of the probability of the points from $\mathbf{X}$ being drawn from the GMMs, by moving the centroids $\mathbf{Y}$ in a coherent manner [27]. This optimization is formulated as an Expectation Maximization (EM) problem.

For non-rigid registration, the deformed point set $\mathcal{T}$ is described by the initial position of the points $\mathbf{Y}$ and a displacement function v:

$$\mathcal{T}(\mathbf{Y}, v) = \mathbf{Y} + v(\mathbf{Y}). \tag{1}$$

For any D-dimensional set of points $\mathbf{Z} \in \mathbb{R}^{N \times D}$, the displacement function v is defined as:

$$v(\mathbf{Z}) = \mathbf{G}(\mathbf{Y}, \mathbf{Z})\mathbf{W}, \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{N \times D}$ is a weight matrix, which can be interpreted as a set of D-dimensional deformation vectors. This matrix $\mathbf{W}$ is estimated in the M-step of the EM algorithm. The coherent movement is controlled by the Gaussian kernel matrix $\mathbf{G}(\mathbf{Y}, \mathbf{Z})$ [19], defined element-wise as:

$$g_{ij} = \mathbf{G}(y_i, z_j) = \exp(-\frac{1}{2\beta^2} \left\| y_i - z_i \right\|^2), \tag{3}$$

where β is a parameter that controls the interaction between points. For convenience in the notation, $\mathbf{G}(\mathbf{Y}, \mathbf{Y})$ is denoted as $\mathbf{G}$. For an in-depth derivation of CPD, please refer to [19].

B. Shape Space

For a known object category, we construct a shape space that describes typical geometrical variations of the category. An object category is defined as a set of 3D object models with similar extrinsic geometry. In order to ensure consistency between the deformations, all the 3D models are aligned to a common coordinate frame. Initially, a canonical model $\mathbf{C}$ is chosen and each instance i is deformed against this canonical model following:

$$\mathcal{T}_i(\mathbf{C}, \mathbf{W}_i) = \mathbf{C} + \mathbf{G}(\mathbf{C}, \mathbf{C})\mathbf{W}_i. \tag{4}$$

The shape space is defined as a low-dimensional space spanned by the principal components of all deformation fields $\mathbf{W}_i$. This is possible based on the observation that the $\mathbf{G}$ matrix only depends on the point set that is deformed,

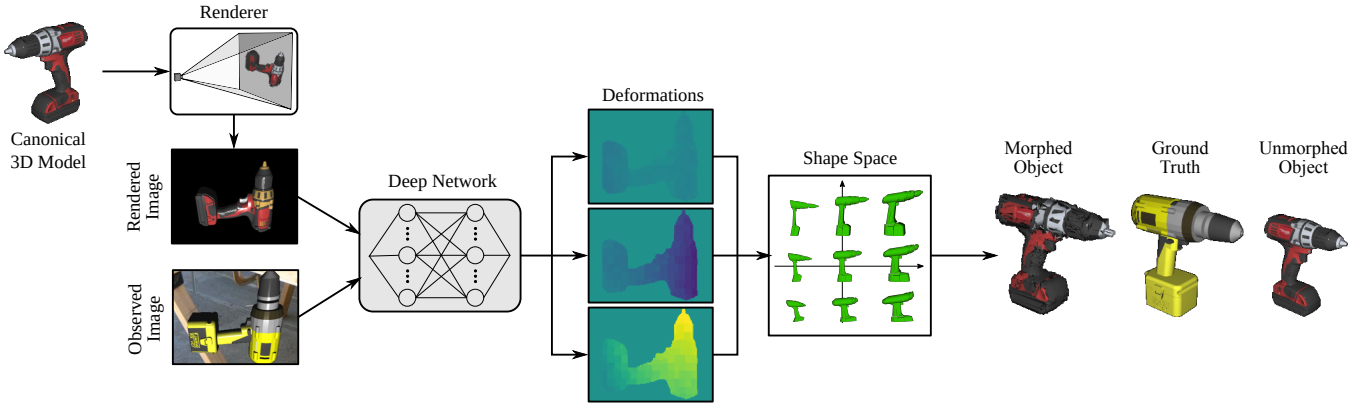

Fig. 2. A canonical model is rendered in the same pose as the observed instance. Then, a deep CNN infers the x, y, and z deformation components for each pixel of the rendered image. The deformations of the occluded parts are inferred by a shape space of the object category. Thanks to the shape space, the object is reconstructed (morphed object), which is shown from a non-observable viewpoint. For comparison, the ground truth object and the unmorphed canonical model are shown at the rightmost.

i.e., the canonical model. In other words, the uniqueness of an object deformation inside a category is fully captured by its $\mathbf{W}_i$ matrix.

Note that the dimensionality of all $\mathbf{W}_i$ matrices is the same. This allows us to define feature vectors $\mathbf{w}_i \in \mathbb{R}^{3n}$, and perform the Principal Component Analysis Expectation Maximization (PCA-EM) to find the latent space of the category. As result, a matrix $\mathbf{L} \in \mathbb{R}^{l \times 3n}$ containing l principle components is determined. Any point $\mathbf{x} \in \mathbb{R}^l$ in the latent space can now be mapped into a feature vector $\hat{\mathbf{w}} \in \mathbb{R}^{3n}$ by:

$$\hat{\mathbf{w}} = \mathbf{L}\mathbf{x} + \bar{\mathbf{w}}, \tag{5}$$

where $\bar{\mathbf{w}}$ is the mean of all feature vectors. In this manner, the canonical matrix $\mathbf{C}$ together with the principle components $\mathbf{L}$ represent the deformation model for an object category.

IV. Method

In this section, we describe our approach that deforms a canonical 3D model of an object category into a novel instance observed by a single RGB image (Fig. 2). Our approach is composed of three main components: a renderer [10], a deep CNN, and a shape space. The renderer is in charge of generating 2D images of realistic 3D models which will be used to train the CNN. Given the images of the canonical and observed models, the CNN infers a three-channel image that represents deformation vectors for each of the visible pixels of the image of the canonical model. The final deformation field is estimated by searching in the shape space for a feature vector whose deformation field matches best the deformations inferred by the network. By means of the shape space, an estimation of the deformation field of the non-observable parts is inferred.

A. Deformation Representation

For a given object category, e.g., *drill*, a canonical model is chosen. This model consists of a textured three-dimensional mesh. Model point clouds can be generated by ray-casting from several viewpoints on a tessellated sphere followed by down-sampling, e.g., by a voxel grid filter. The matrix of the point cloud of the canonical model is referred as $\mathbf{C} \in \mathbb{R}^{n \times 3}$, while the mesh point matrix is denoted $\mathbf{C}_m \in \mathbb{R}^{m \times 3}$. The point cloud matrix is used to define a deformation on the mesh. The matrix of the deformed mesh $\mathbf{C}'_m \in \mathbb{R}^{m \times 3}$ is defined as:

$$\mathbf{C}'_m = \mathbf{C}_m + \mathbf{G}(\mathbf{C}_m, \mathbf{C})\mathbf{W}(\mathbf{C}, \mathbf{O}), \tag{6}$$

where $\mathbf{W}(\mathbf{C}, \mathbf{O}) \in \mathbb{R}^{n \times 3}$ describes the offsets that should be applied to the points of the canonical point cloud $\mathbf{C}$ to deform it towards the point cloud of the observed instance represented by $\mathbf{O} \in \mathbb{R}^{k \times 3}$. The offsets $\mathbf{W}(\mathbf{C}, \mathbf{O})$ are multiplied by $\mathbf{G}(\mathbf{C}_m, \mathbf{C}) \in \mathbb{R}^{m \times n}$ to map them into the offsets of the mesh vertices. $\mathbf{G}(\mathbf{C}_m, \mathbf{C})$ is calculated as described in Eq. (3) and ensures coherent movement of the vertices. Note that $\mathbf{W}(\mathbf{C}, \mathbf{O})$ is the only part of Eq. (6) that depends on the observed instance.

To generate target images for training the CNN, we define the offset matrix $\boldsymbol{\delta}(\mathbf{C}, \mathbf{O}) \in \mathbb{R}^{n \times 3}$:

$$\boldsymbol{\delta}(\mathbf{C}, \mathbf{O}) = \mathbf{G}(\mathbf{C}, \mathbf{C})\mathbf{W}(\mathbf{C}, \mathbf{O}). \tag{7}$$

and represent it as a three channel image, where each channel describes the deformations in one of the coordinate axis: x, y and z. The mapping from the 3D offset matrix δ to the image space is described in Sec. V.

B. Zoom Operation

Inspired by [16], the amount of object details is increased by zooming in the observed and rendered images before feeding them into the network. Given the pose of the observed object, the canonical model is rendered placing the object at the same place as the observed one with respect to the camera. Then, a bounding box that contains both objects and has the same aspect ratio as the input images of the network is defined. Following this bounding box, both images are cropped and upsampled bilinearly to the fixed size of the network input (256×192 in our experiments). An example of the zoom operation is shown in Figure 3.

C. Convolutional Neural Network

The backbone of our network is the FlowNet2 architecture [28] because we assume a connection between the

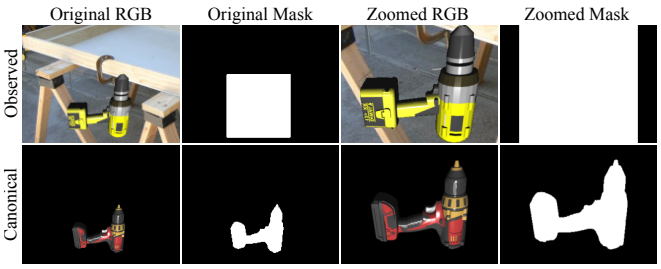

Fig. 3. Zoom operation. The canonical model is rendered at the same pose as the observed one. Both images are cropped according to a bounding box that contains both objects and has the aspect ratio of the network input. Finally, the cropped images are bilinear upsampled.

optical flow among two objects and the point offsets that should be applied for a matching deformation. We modify FlowNet2 such that it receives two RGB and two mask images as input, and it returns a three channel image representing the deformation for each pixel. The mask image of the canonical model is directly given by the renderer and serves as a regularizer to distinguish the foreground and background pixels while the mask of the observed image is a bounding box containing the object. The mask images can be interpreted as a fourth channel additional to the RGB channels of the two input images. Thus, we use the terms observed and canonical images in the remaining sections of this paper to refer to the network input.

D. Deformation Inference of Occluded Parts

Given a single RGB image, an object cannot be fully observed. Thus, we first find deformations on the visible object pixels and use these deformations to infer the missing ones of the occluded parts. Estimating deformations on the observable parts is handled by the network (Sec.IV-C). For inferring deformations of the occluded parts, a shape space is constructed as explained in Sec. III-B. Then, we search for a feature vector $\hat{\mathbf{w}}$ in the shape (latent) space that matches best the deformations of the observable points.

The CNN outputs a three channel image containing a deformation vector per pixel. Now, a mapping from the image space to the vector space of $\mathbf{C}$ needs to be established. Thus, we find the closest point on $\mathbf{C}$ to the position of each pixel in the position tensor given by the renderer. In general, the number of pixels is larger than the number of points of $\mathbf{C}$, and multiple pixels are assigned to the same point of $\mathbf{C}$. Consequently, the deformation of the closest points of $\mathbf{C}$ is defined as the mean of the assigned deformations. This results in a matrix that describes how the closest points of the canonical model have to be moved to match the shape of the visible parts of the observed object. For the occluded points, we assume an initial offset of zero and build a sparse matrix $\boldsymbol{\delta}_{vis} \in \mathbb{R}^{n\times 3}$ with the deformation of the closest points.

Obtaining a deformation vector for all points of $\mathbf{C}$ can be formulated as the task of searching for a feature vector $\mathbf{x}$ in the shape space that matches the inferred deformations of the visible points. This feature vector $\mathbf{x}$ defines a deformation matrix:

$$\hat{\boldsymbol{\delta}} = \tilde{\mathbf{G}}(\mathbf{C},\mathbf{C})(\mathbf{L}\mathbf{x} + \bar{\mathbf{w}}), \tag{8}$$

where $\tilde{\mathbf{G}}(\mathbf{C},\mathbf{C}) \in \mathbb{R}^{3n\times 3n}$ is a rearranged form of $\mathbf{G}(\mathbf{C},\mathbf{C}) \in \mathbb{R}^{n\times n}$ with additional zeros to match the dimensionality of $\mathbf{L}\mathbf{x}+\bar{\mathbf{w}} \in \mathbb{R}^{3n\times 1}$. Our objective is then to minimize the loss on the visible points between $\boldsymbol{\delta}_{vis}$ and $\hat{\boldsymbol{\delta}}$:

$$L(\boldsymbol{\delta}_{vis}, \hat{\boldsymbol{\delta}}) = \sum_{v\in i_{vis}} \left\| \boldsymbol{\delta}_{vis}(v,.) - \hat{\boldsymbol{\delta}}(v,.) \right\|^2, \tag{9}$$

where i_{vis} describes the set of indices belonging to visible points and $\boldsymbol{\delta}(v,.)$ describes the v-th row of matrix $\boldsymbol{\delta}$. This minimization problem can be expressed as a least squares problem. Let n_v denote the number of visible points and let define $\overline{\boldsymbol{\delta}}_{vis} \in \mathbb{R}^{3n_v}$ as $\boldsymbol{\delta}_{vis}$ after removing every row containing zeros (occluded points) and rearranging it as a row vector. We introduce a matrix $\mathbf{D} \in \mathbb{R}^{n_v\times n}$ which removes exactly the same rows from $\hat{\boldsymbol{\delta}}$ as the ones removed from $\boldsymbol{\delta}_{vis}$. In this manner, minimizing Eq. (9) results in:

$$L(\overline{\boldsymbol{\delta}}_{vis}, \mathbf{x}) = \left\| \overline{\boldsymbol{\delta}}_{vis} - \mathbf{D}(\tilde{\mathbf{G}}(\mathbf{C},\mathbf{C})(\mathbf{L}\mathbf{x}+\bar{\mathbf{w}})) \right\|^2. \tag{10}$$

Defining:

$$\mathbf{A} := (\mathbf{D}\tilde{\mathbf{G}}(\mathbf{C},\mathbf{C})\mathbf{L}), \tag{11}$$

and

$$\mathbf{B} := (\overline{\boldsymbol{\delta}}_{vis} - \mathbf{D}\tilde{\mathbf{G}}(\mathbf{C},\mathbf{C})\bar{\mathbf{w}}), \tag{12}$$

the feature vector $\mathbf{x}^*$ that represents the deformation of all points in $\mathbf{C}$ is found by:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} L(\overline{\boldsymbol{\delta}}_{vis}, \mathbf{x}) = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{B}\|^2. \tag{13}$$

Eq. (13) can now be solved by an off-the-shelf linear solver. By using Eq. (5) with $\mathbf{x}^*$ and the corresponding rearrangement, the final deformation field $\hat{\mathbf{W}}^*$ is obtained. Moreover, by using Eq. (6), the observed model is reconstructed. A schematic overview of the deformation inference of occluded parts can be seen in Figure 4.

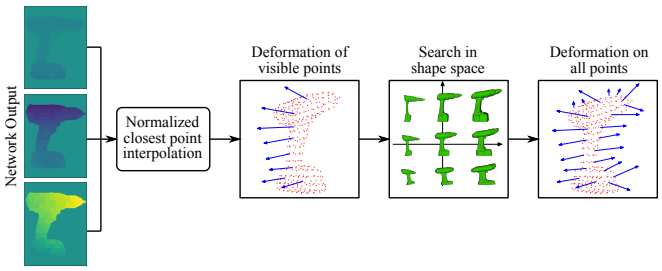

Fig. 4. The network output defines a deformation field for a 3D point set generated by a normalized closest point interpolation. This point set represents the visible points of the canonical model. Deformation vectors for all points of $\mathbf{C}$ are inferred by means of the shape space of the category.

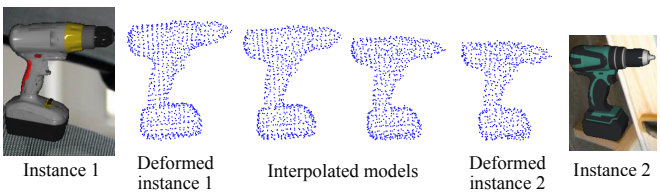

Fig. 5. Registration of two observed instances against each other. The resulting deformed models are interpolated for visualization.

Fig. 6. Samples of 3D models of the *spray bottle*, *camera* and *drill*, and *bottles* categories used in our experiments. The canonical models are leftmost.

Note that two observed instances can be registered against each other by simply finding their respective deformation fields towards the canonical model, since the dimensionality of the deformed objects is defined only by the shared canonical model (Fig. 5).

V. Dataset

To the best of our knowledge, there does not exist an available dataset that contains intra-class object deformations suitable for training CNNs. In this section, we explain how the dataset is generated. Our dataset is composed of a set of 3D textured meshes that produce a set of rendered images and ground truth deformations for each object category.

To build the dataset, we collected realistic textured 3D meshes of different objects from the same object category. The models were taken from online databases[2] and from [29]. A canonical model is selected and separated from the other k models. This canonical model and the collection of 3D data is also used for building the latent space for inferring deformations of the occluded parts (Sec. IV-D). Ground truth deformations from the canonical model towards all the other models are calculated using CPD (Sec. III-A). The objects are observed from different viewpoints to generate the rendered images.

Since good quality 3D textured models are scarce, specially graspable objects, we interpolate models that are in between the canonical and the observed models (Fig. 7). To achieve this, $\mathcal{T}(\mathbf{C}, \mathbf{W}_i)$ is calculated for all the testing models, as described in Eq. (4). Note that simply using Eq. (6) to interpolate from the canonical and to the observed models will generate instances with the same texture as the canonical mesh, reducing the generalization capabilities of the network. Therefore, an inverse deformation (from observed to canonical) is defined based on the point cloud of the observed training instance $\mathbf{T}_i$ and the deformations fields $\mathbf{W}_i$:

$$\mathcal{T}^{-1}(\mathbf{T}_i, \mathbf{W}_i) = \mathbf{T}_i + \mathbf{G}(\mathbf{T}_i, \mathbf{C})(-\mathbf{W}_i). \quad (14)$$

Similarly to Eq. (6), the vertices $\mathbf{T}_{m,i}$ of the observed training instances are morphed following:

$$\mathbf{T}'_{m,i} = \mathbf{T}_{m,i} + \mathbf{G}(\mathbf{T}_{m,i}, \mathbf{C})(-\mathbf{W}_i). \quad (15)$$

Finally, by adding an interpolation factor ρ we obtain:

$$\mathbf{T}'_{m,i}(\rho) = \mathbf{T}_{m,i} + \mathbf{G}(\mathbf{T}_{m,i}, \mathbf{C})(-\rho\mathbf{W}_i). \quad (16)$$

Eq. (16) allows us to interpolate between every observed model and the canonical model without incurring in expensive computations of CPD for each interpolation model.

For a given observed model i and a given interpolation constant ρ, we morph the model according to Eq. (16). This morphed model and the canonical model are rendered from different viewpoints. We generate a bounding box for the image of the observed model and a mask for the rendered canonical image. This four images are then zoomed in as explained in Sec. IV-B.

Apart from the RGB image, the renderer also provides a three channel position image. This image contains in each

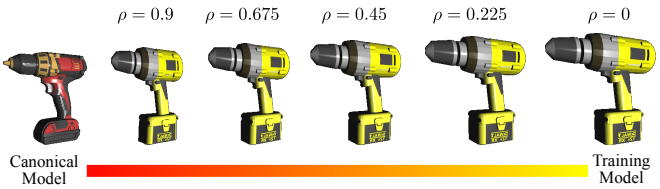

Fig. 7. Interpolation process from the observed instance on the right towards the canonical instance on the left.

[2] https://sketchfab.com

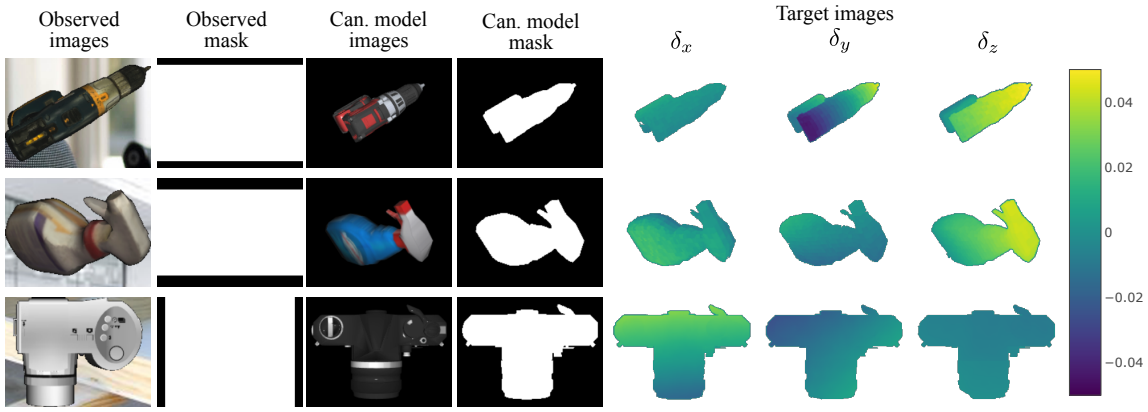

Fig. 8. Generated training images. The networks receive four images: an observed RGB image with its bounding box mask image, and a rendered image of the canonical model with its mask image. The target is a three channel image containing the x, y and z component of the deformations which are represented using heatmaps. For clarity in the visualization, the background of the target images are painted white.

pixel the information about the x, y, and z coordinates of the visible parts of the rendered model. The position image is cropped and upsampled in the same way as the other four images. The resulting position tensor defines a 3D point set $\mathbf{P}$. Given the canonical point set and the corresponding deformation vector for each point, we perform a radial basis function interpolation with a linear kernel to find the deformation vector for each point of $\mathbf{P}$. The results of the interpolation are then expressed as a three channel image, each channel containing the x, y and z deformations for each pixel. The ground truth target data for the network is generated as explained in Sec. IV-A. However, because of the interpolated models, Eq. 7 is expanded as:

$$\boldsymbol{\delta}(\mathbf{C}, \mathbf{T}_i) = \mathbf{G}(\mathbf{C}, \mathbf{C})(1-\rho)\mathbf{W}_i. \tag{17}$$

VI. Evaluation

A. Experimental Setup

We tested our approach on four categories: *spray bottles*, *cameras*, *bottles*, and *drills* containing 12, 15, 14 and 16 instances, respectively. Figure 6 shows 3D model samples of the categories used in the experiments. For the dataset generation, we used four $\rho \in \{0.0, 0.25, 0.5, 0.75\}$ interpolation values and 74 viewpoints on a tessellated sphere. Pulling apart one instance as the canonical model and two instances for testing results in 2664, 3552, 3256 and 3848 training images, respectively. To obtain ground truth deformations we used CPD with $\lambda = 2.0$ and $\beta = 2.0$ across all object instances. All the shape spaces have $l = 5$ latent dimensions. The testing instances (T1 and T2 for each category) are not used—neither for training the network nor for building the shape spaces. In this manner, the testing instances are completely novel when presented to our approach for evaluation. Samples of the generated training images are shown in Figure 8. Ground truth object poses are used in order to evaluate only the performance of the non-rigid registration. The robustness of our method is however evaluated against noise on the object pose. Note that object poses can be estimated using approaches such as [30] or [12].

The training images are split up randomly. 90% are used for actual training and 10% are used for validation. For training, the ground truth deformation are scaled by a factor of 1000, to make a clear boundary between foreground and background. The L_2 loss is used for training the network. We trained on two graphic cards with a batch size of 24 each (effective batch size of 48). The learning rate is decreased stepwise, starting from 3×10^{-5} it is divided by two every 600 epochs, until reaching a final learning rate of 1×10^{-6}. The training is done for 2000 epochs.

B. Experimental Results

To evaluate the quality of the registration, the following error function between two point clouds $\mathbf{C} \in \mathbb{R}^{n\times 3}$ and $\mathbf{T} \in \mathbb{R}^{m\times 3}$ is defined:

$$E(\mathbf{T}, \mathbf{C}) = \frac{1}{m}\sum_{i=0}^{m-1} \min_j \|\mathbf{T}(i,.) - \mathbf{C}(j,.)\|^2. \tag{18}$$

It computes the mean distance between the points of the observed model to the respective closest point of the deformed canonical point cloud.

We compare our results against the Categorical Latent Space (CLS) approach [20] and CPD [19]. Note, however, that those approaches require 3D data in contrast to our

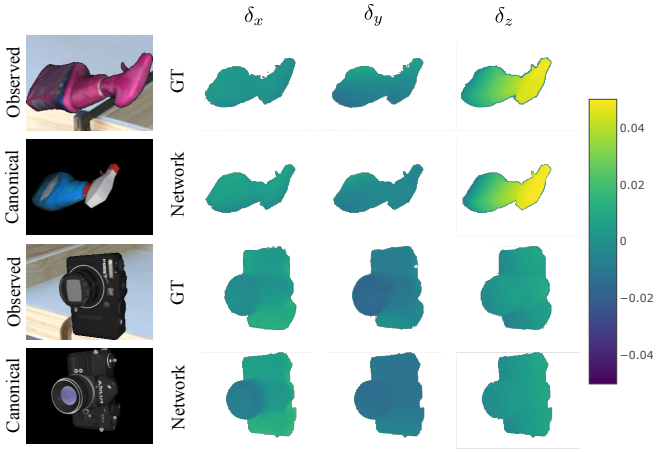

Fig. 9. Estimated deformations (in meters) of novel instances presented to the network. For comparison, the ground truth (GT) is also shown. For clarity in the visualization, background pixels are painted white.

method. The latent space approach [20] and the CPD method are parameterized using the same values as for generating the training images. This encourages a fair evaluation, because a bad parametrization of CPD will affect the quality of our training images and shape space. The latent space dimensionality of CLS is also set to five. To generate testing images, we rendered the testing instances from 74 points of view on a tessellated sphere and produced a point cloud to represent the observed parts of the instances. In addition, the

TABLE I. Comparison with other approaches.
Mean and (standard deviation) error values expressed in μm.

Instance	Ground Truth	CLS [20] (3D data)	CPD [19] (3D data)	Ours (RGB)
Camera T1	34.61 (1.97)	51.93 (10.45)	407.04 (491.89)	102.17 (47.89)
Camera T2	16.45 (1.61)	19.87 (4.59)	167.18 (358.16)	18.80 (5.11)
Bottle T1	23.25 (2.34)	25.92 (5.18)	140.51 (312.13)	45.21 (9.75)
Bottle T2	90.42 (28.54)	72.33 (11.35)	357.98 (536.81)	88.35 (18.39)
Sp. Bottle T1	29.84 (1.42)	30.78 (1.89)	298.53 (409.14)	47.87 (12.99)
Sp. Bottle T2	111.94 (14.29)	121.19 (19.16)	376.09 (720.73)	154.97 (82.34.39)
Drill T1	21.18 (0.949)	28.86 (1.42)	232.88 (1319)	52.71 (23.54)
Drill T2	63.95 (5.23)	58.50 (21.51)	216.35 (566.18)	119.88 (107.43)

deformation that results from the ground truth target images is calculated. This establishes a bound for the results of our approach and allows us to evaluate the network performance. Figure 9 shows the ground truth target deformations and the network output for two instances of the spray and camera categories. The mean and standard deviations of the error values of all four categories are presented in Table I. The CLS approach [20] performs best overall, since it also incorporates a shape space and had in contrast to our approach access to 3D data. Our approach outperforms CPD by a large margin even without access to 3D data for the evaluated categories. Compared to CPD, the lower error mean indicates that our approach reconstructs the objects better while the lower standard deviation demonstrates that out method is more robust to different object viewpoints.

We further evaluated our approach against noise on the

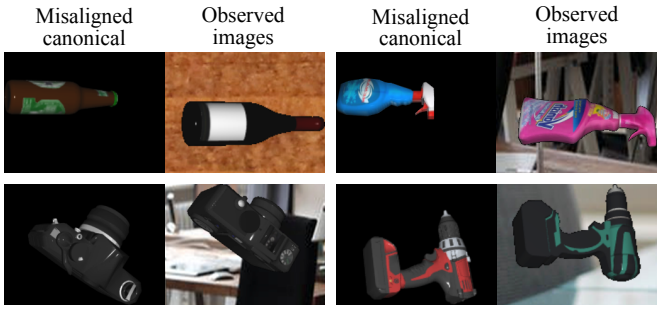

Fig. 10. Network input images after noise on the object pose is applied.

TABLE II. Evaluation against noise on the object pose.
Mean and (standard deviation) error values expressed in μm.

Category	CPD [19] (3D data)	Ours (RGB)
Camera T1	168.54 (357.8)	105.26 (64.21)
Camera T2	406.45 (492.03)	306.96 (127.89)
Bottle T1	297.79 (579.49)	227.90 (146.0)
Bottle T2	852.40 (1818)	289.36 (147.68)
Spray Bottle T1	1035 (406.69)	146.89 (117.57)
Spray Bottle T2	1488 (554.33)	255.69 (167.32)
Drill T1	232.35 (1325)	92.96 (58.23)
Drill T2	215.54 (565.48)	262.31 (228.40)

given object pose. A three-dimensional vector is uniformly sampled from $[-0.05, 0.05]$ for each coordinate axis and added to the canonical object pose. Resulting rendered images are shown in Figure 10. We compare our performance against CPD; the results are presented in Table II. As expected, the performance of our approach is affected by this large misalignment, because no pose refinement is explicitly modeled. However, we achieved lower registration errors compared to CPD.

We finally evaluated the performance of our approach with real object images. Figure 11 shows the input to the network and the resulting deformed models. The applicability of our method on real images is demonstrated based on the plausible registration results. The reconstruction of the white spray bottle does not match well with the ground truth possibly because none of the training instances exhibits such large

Fig. 11. Non-rigid registration on real pictures. The ground truth of two objects was available and is presented next to the reconstructed object. Hard objects to perceive as transparent bottles are successfully registered.

TABLE III. Evaluation with real images.
Mean and (standard deviation) error values expressed in μm.

Category	CPD [19] (3D data)	Ours (RGB)
Real Spray B. (pink)	451.78 (329.7)	67.34 (41.18)
Real Spray B. (white)	436.38 (409.25)	398.03 (229.05)

dimensions. The ground truth 3D models were available for two real spray bottles. We computed the registration error for 12 different object poses by turning these two real objects around their vertical axis from their standing position every 30°. The results are presented in Table III.

VII. Conclusion

We presented a novel approach for category-level non-rigid registration based on single-view RGB images. We demonstrated that a neural network is able to infer deformations on the visible parts of the observed object. In addition, we showed how objects are reconstructed by incorporating a learned shape space for each category. We evaluated our approach on synthetic and real images outperforming CPD even without depth information and with noise on the object pose. Furthermore, we demonstrated that objects which are hard-to-measure by depth sensors (e.g., transparent bottles) can by successfully registered with our method.

In the future, we plan to extend our neural network in order to infer pixel-wise object categories to better match the masks used for training. Furthermore, we will incorporate an object pose refinement module on the network and the shape space to increase robustness against misalignments.

References

[1] D. Pavlichenko, D. Rodriguez, C. Lenz, M. Schwarz, and S. Behnke, "Autonomous bimanual functional regrasping of novel object class instances," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2019.

[2] D. Pavlichenko, D. Rodriguez, M. Schwarz, C. Lenz, A. S. Periyasamy, and S. Behnke, "Autonomous Dual-Arm Manipulation of Familiar Objects," in *IEEE-RAS 18th Int. Conf. on Humanoid Robots (Humanoids)*, 2018.

[3] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel, "A statistical model of human pose and body shape," in *Computer Graphics Forum (Eurographics 2009)*, 2009.

[4] B. Allen, B. Curless, and Z. Popović, "The space of human body shapes: reconstruction and parameterization from range scans," *ACM Transactions on Graphics (TOG)*, 2003.

[5] T. Stouraitis, U. Hillenbrand, and M. A. Roa, "Functional power grasps transferred through warping and replanning," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[6] D. Rodriguez, A. Di Guardo, A. Frisoli, and S. Behnke, "Learning Postural Synergies for Categorical Grasping through Shape Space Registration," in *IEEE-RAS 18th Int. Conf. on Humanoid Robots (Humanoids)*, 2018.

[7] B. J. Brown and S. Rusinkiewicz, "Global non-rigid alignment of 3D scans," *ACM Transactions on Graphics (TOG)*, 2007.

[8] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, *et al.*, "Real-time non-rigid reconstruction using an RGB-D camera," *ACM Transactions on Graphics (TOG)*, 2014.

[9] D. Rodriguez and S. Behnke, "Transferring category-based functional grasping skills by latent space non-rigid registration," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 2662–2669, 2018.

[10] M. Schwarz and S. Behnke, "Stillleben: Realistic scene synthesis for deep learning in robotics," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020.

[11] J. Tremblay, T. To, and S. Birchfield, "Falling things: A synthetic dataset for 3D object detection and pose estimation," in *IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, 2018.

[12] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Robotics: Science and Systems (RSS)*, 2018.

[13] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser, "Physically-based rendering for indoor scene understanding using convolutional neural networks," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[15] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Conf. on Robot Learning (CoRL)*, 2018.

[16] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIm: Deep iterative matching for 6D pose estimation," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[17] V. G. Kim, Y. Lipman, and T. Funkhouser, "Blended intrinsic maps," in *ACM Transactions on Graphics (TOG)*, 2011.

[18] Y. Zeng, C. Wang, Y. Wang, X. Gu, D. Samaras, and N. Paragios, "Dense non-rigid surface registration using high-order graph matching," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[19] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2010.

[20] D. Rodriguez, C. Cogswell, S. Koo, and S. Behnke, "Transferring grasping skills to novel instances by latent space non-rigid registration," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[21] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *28th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2001, pp. 67–76.

[22] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (TOG)*, 2013.

[23] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *4th Int. Conf. on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (GRAPHITE)*, 2006.

[24] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," in *European Conf. on Computer Vision (ECCV)*, 2016.

[25] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem, "Completing 3D object shape from one depth image," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[26] J. Wu, C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum, "Learning shape priors for single-view 3D completion and reconstruction," in *European Conf. on Computer Vision (ECCV)*, 2018.

[27] A Yuille, "The motion coherence theory," in *Int. Conf. on Computer Vision (ICCV)*, 1998.

[28] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6D object pose and size estimation," in *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] C. Capellen, M. Schwarz, and S. Behnke, "ConvPoseCNN: Dense convolutional 6D object pose estimation," in *15th Int. Conf. on Computer Vision Theory and Applications (VISAPP)*, 2020.