Domain Transfer for 3D Pose Estimation from Color Images without Manual Annotations

Mahdi Rad[1 [0000−0002−4011−4729]], Markus Oberweger[1 [0000−0003−4247−2818]], and Vincent Lepetit[2,1 [0000−0001−9985−4433]]

[1] Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria

[2] Laboratoire Bordelais de Recherche en Informatique, Université de Bordeaux, Bordeaux, France

{rad,oberweger,lepetit}@icg.tugraz.at

Abstract. We introduce a novel learning method for 3D pose estimation from color images. While acquiring annotations for color images is a difficult task, our approach circumvents this problem by learning a mapping from paired color and depth images captured with an RGB-D camera. We jointly learn the pose from synthetic depth images that are easy to generate, and learn to align these synthetic depth images with the real depth images. We show our approach for the task of 3D hand pose estimation and 3D object pose estimation, both from color images only. Our method achieves performances comparable to state-of-the-art methods on popular benchmark datasets, without requiring any annotations for the color images.

Keywords: Domain transfer · 3D object pose estimation · 3D hand pose estimation · Synthetic data.

1 Introduction

3D pose estimation is an important problem with many potential applications. Recently, Deep Learning methods have demonstrated great performance, when a large amount of training data is available [1,2,3,4,5]. To create training data, the labeling is usually done with the help of markers [6,7] or a robotic system [8], which in both cases is very cumbersome, expensive, or sometimes even impossible, especially from color images. For example, markers cannot be used for 3D hand labeling of color images, as they change the appearance of the hand.

Another direction is to use synthetic images for training. However, synthetic images do not exactly look like real images. Generative Adversarial Networks (GANs) [9,10,11,12] or transfer learning techniques [13,14,15,16] can be used to bridge the domain gap between real and synthetic images. However, these approaches still require some annotated real images to learn the domain transfer. [2] relies on registered real images to compute a direct mapping between the image features of real and synthetic images, but it also requires some labeled real images.

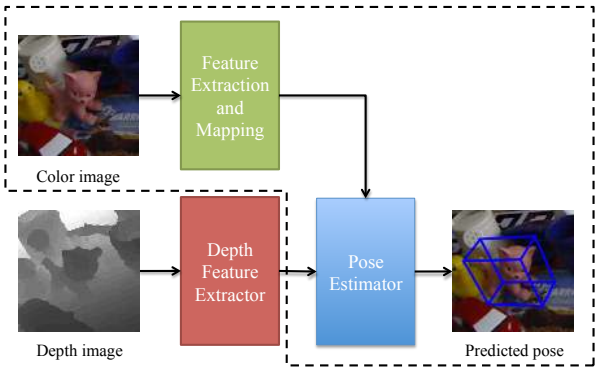


Fig. 1: Method overview. We train a depth feature extractor (red box) together with a pose estimator (blue box). We also train a second network (green box), which extracts image color features and maps them to the depth space, given color images and their corresponding depth images. At run-time, given a color image, we map color features to depth space in order to use the pose estimator to predict the 3D pose of the object (dashed lines). This removes the need for labeled color images.

In this paper, we propose a method that learns to predict a 3D pose from color images, without requiring labeled color images. Instead, it exploits labeled depth images. These depth images can be real depth images, which are easier to label than color images, and are already readily available for some problems. More interestingly, they can also be synthetic depth images: Compared to color images, synthetic depth images are easier to render, as there is no texture or illumination present in these images.

An overview of our approach is shown in Fig. 1. Our main idea is to bridge the domain gap between color images and these synthetic depth images in two steps, each one solving an easier problem than the original one. We use an RGB-D camera to capture a set of pairs made of color and depth images that correspond to the same view. Capturing such a set can be done by simply moving the camera around. We apply [2] to this set and learn to map the features from the color images to corresponding depth images. However, this mapping alone is not sufficient: A domain gap between the depth images captured by the RGB-D camera and the available labeled depth images remains, since the labeled depth images could be captured with another RGB-D camera or rendered synthetically. Fortunately, this remaining gap is easier to bridge than the domain gap between real and synthetic color images, since illumination and texture effects are not present in depth images. To handle it, we use Maximum Mean Discrepancy (MMD) [17] to measure and minimize the distance between the means of the features of the real and synthetic depth images mapped into a Reproducing Kernel Hilbert Space (RKHS). MMD is a popular in domain transfer method [16] since it does

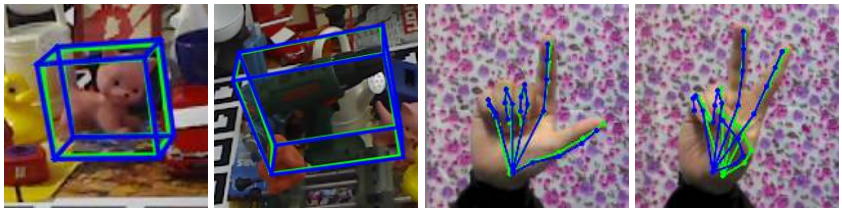


Fig. 2: Our method allows very accurate 3D pose estimation from color images without annotated color images. In case of 3D rigid object pose estimation we draw the bounding boxes, where blue is the ground truth bounding box and red the bounding box of our predicted pose. For 3D hand pose estimation, we show the 3D joint locations projected to the color image, blue denoting the ground truth and green our estimation.

not require correspondences to align the features of different domains and can be efficiently implemented.

Our approach is general, and not limited to rigid objects. It can be applied to many other applications, such as 3D hand pose estimation, human pose estimation, etc. Furthermore, in contrast to color rendering, no prior information about object's texture has to be known. Fig. 2 shows applications to two different problems: 3D rigid object pose estimation and 3D hand pose estimation from color images, on the LINEMOD [6] and STB [18] datasets, respectively. Our method achieves performance comparable to state-of-the-art methods on these datasets without requiring any annotations for the color images.

In the remainder of this paper, we discuss related work, then present our approach and its evaluation.

2 Related Work

We first review relevant works for 3D pose estimation from color images, and then review related methods on domain transfer learning.

2.1 3D Pose Estimation from Color Images

Inferring the 3D pose from depth images has achieved excellent results [1,5,19,20], however, inferring the 3D pose from color images still remains challenging. [4] presented an approach for 3D object pose estimation from color images by predicting the 2D locations of the object corners and using PnP to infer the 3D pose, similar to [3,21]. Also, [22] first predicts the 2D joint locations for hand pose estimation, and then lifts these prediction to 3D estimates. [23] predicts 2D and 3D joint locations jointly, and then applies inverse kinematics to align these predictions. Similarly, [24] uses inverse kinematics to lift predicted 2D joint locations to 3D. All these approaches are fully supervised and require annotated color

images, which are cumbersome to acquire in practice. Recently, [25] uses synthetically generated color images from 3D object models with pretrained features, however, they require extensive refinement of the initial network predictions, and we will show that we can reach better performances without annotations for real color images when using no refinement. To generalize synthetically generated color images to real images, [26] proposed to use a domain randomization method, however, the generalization is still limited, and outperformed by our approach as we show in the Evaluation section.

2.2 Domain Transfer Learning

As we mentioned in the introduction, it is difficult to acquire annotations for real training data, and training on synthetic data leads to poor results [4,25]. This is an indication for a domain gap between synthetic and real training data. Moreover, using synthetic data still requires accurately textured models [4,8,25,27] that require large amount of engineering to model. On the other hand, synthetic depth data is much simpler to produce, but still it requires a method for domain transfer.

A popular method is to align the distributions for the extracted features from the different domains. Generative Adversarial Networks (GANs) [10] and Variational Autoencoders (VAEs) [28] can be used to learn a common embedding for the different domains. This usually involves learning a transformation of the data such that the distributions match in a common subspace [13,14,15]. [29] learns a shared embedding of images and 3D poses, but it requires annotations for the images to learn this mapping. Although GANs are able to generate visually similar images between different domains [12], the synthesized images lack precision required to train 3D pose estimation methods [2,9]. Therefore, [23] developed a sophisticated GAN approach to adapt the visual appearance of synthetically rendered images to real images, but this still requires renderings of high-quality synthetic color images.

To bridge this domain gap, [2] predicts synthetic features from real features and use these predicted features for inference, but this works only for a single modality, *i.e.* depth or color images, and requires annotations from both modalities. Similarly, [30] transfers supervision between images from different modalities by learning representations from a large labeled modality as a supervisory signal for training representations for a new unlabeled paired modality. In our case, however, we have an additional domain gap between real and synthetic depth data, which is not considered in their work. Also, [31] aims at transforming the source features into the space of target features by optimizing an adversarial loss. However, they have only demonstrated this for classification, and this approach works poorly for regression [2]. [16] proposed a Siamese Network for domain adaptation, but instead of sharing the weights between the two streams, their method allows the weights to differ and only regularizes them to keep them related. However, it has been shown in [2] that the adapted features are not accurate enough for 3D pose estimation.

Differently, [32] transfers real depth images to clean synthetic-looking depth images. However, this requires extensive hand-crafted depth image augmentation to create artificial real-looking depth images during training, and modeling the noise of real depth cameras is difficult in practice. [33] proposed to randomize the appearance of objects and rendering parameters during training, in order to improve generalization of the trained model to real-world scenarios. However, this requires significant engineering effort and there is no guarantee that these randomized renderings cover all the visual appearances of a real-world scene.

Several works [34,35] propose a fusion of features from different domains. [34] fuses color and depth features by using labeled depth images for a few categories and adapts a color object detector for a new category such that it can use depth images in addition to color images at test time. [35] propose a combination method that selects discriminative combinations of layers from the different source models and target modalities and sums the features before feeding them to a classifier. However, both works require annotated images in all domains. [36] uses a shared network that utilizes one modality in both source and target domain as a bridge between the two modalities, and an additional network that preserves the cross-modal semantic correlation in the target domain. However, they require annotations in both domains, whereas we only require annotations in one, *i.e.* the synthetic, domain that are much easier to acquire.

When comparing our work to these related works on domain transfer learning, these methods either require annotated examples in the target domain [23,36,34,35,2,31], are restricted to two domains [30,35,34,2,31], or require significant engineering [32,33]. By contrast, our method does not require any annotations in the target domain, *i.e.* color images, and can be only trained on synthetically rendered depth images that are easy to generate, and the domain transfer is trained from real data that can be easily acquired using a commodity RGB-D camera.

3 Method

Given a 3D model of a target object, it is easy to generate a training set made of many depth images of the object under different 3D poses. Alternatively, we can use an existing dataset of labeled depth images. We use this training set to train a first network to extract features from such depth images, and a second network, the *pose estimator*, to predict the 3D pose of an object, given the features extracted from a depth image. Because it is trained on many images, the pose estimator performs well, but only on depth images.

To apply the pose estimator network to color images, we train a network to map features extracted from color images to features extracted from depth images, as was done in [2] between real and synthetic images. To learn this mapping, we capture a set of pairs of color and depth images that correspond to the same view, using an RGB-D camera. In order to handle the domain gap between the real and synthetic depth images of two training sets, we apply the Maximum Mean Discrepancy (MMD) [17], which aims to map features of each training set to a Reproducing Kernel Hilbert Space (RKHS) and then minimizes

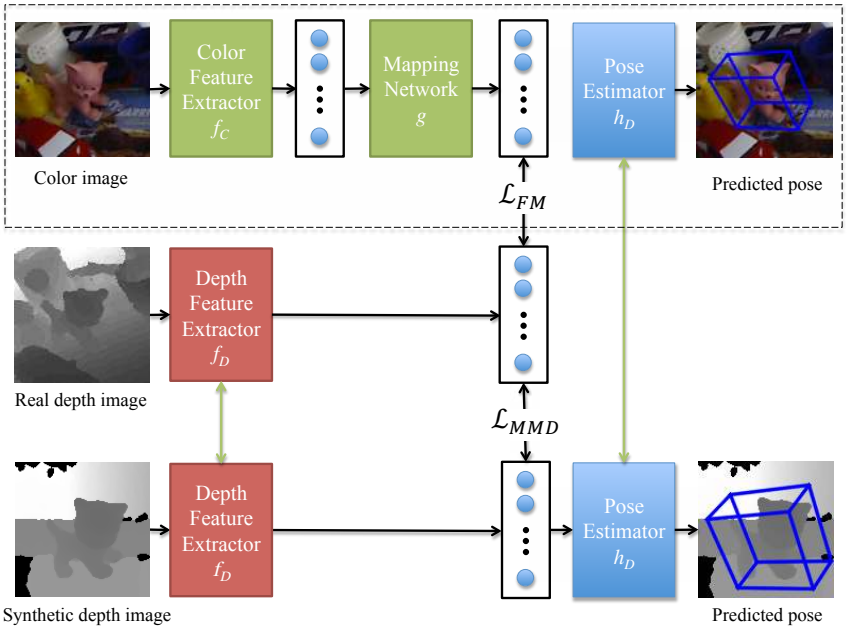


Fig. 3: Detailed overview of our approach. It consists of three data streams, one for each domain. The lower two streams take depth images as input, *i.e.* synthetic and real depth images, respectively, and extract features using the network $f_{\mathcal{D}}$. The upper stream takes color images as input and uses the mapping network g to map the color features to the depth features used for pose prediction with network $h_{\mathcal{D}}$. The parameters of the depth feature extractor $f_{\mathcal{D}}$ and pose predictor $h_{\mathcal{D}}$ are shared between the synthetic and real depth image (green arrows). Between the synthetic and the real depth feature we have the MMD loss $\mathcal{L}_{MMD}$ and between the real color and real depth features we use the feature mapping loss $\mathcal{L}_{FM}$. For inference at test time, we use only the upper stream within the dashed lines that takes a real color image as input, extracts features using the network $f_{\mathcal{C}}$, maps these features to the depth space using g, and uses the pose estimator $h_{\mathcal{D}}$ to predict the 3D pose.

the distance between the means of the mapped features of the two training sets. An overview of the proposed method is shown in Fig. 3.

3.1 Learning the Mapping

More formally, let $\mathcal{T}^{\mathcal{S}} = \{(\mathbf{x}_i^{\mathcal{S}}, \mathbf{y}_i)\}_i$ be a training set of synthetically rendered depth images $\mathbf{x}_i^{\mathcal{S}}$ using a 3D renderer engine under 3D poses $\mathbf{y}_i$. A second training set $\mathcal{T}^{\texttt{RGB-D}} = \{(\mathbf{x}_i^{\mathcal{R}}, \mathbf{x}_i^{\mathcal{D}})\}_i$ consists of pairs of color images $\mathbf{x}_i^{\mathcal{R}}$, and their corresponding depth images $\mathbf{x}_i^{\mathcal{D}}$. We jointly train four networks: the feature extractor for depth images $f_{\mathcal{D}}$, the feature extractor for color images $f_{\mathcal{C}}$, the pose

estimator $h_{\mathcal{D}}$, and the feature mapping network g, on the training sets $\mathcal{T}^{\mathcal{D}}$ and $\mathcal{T}^{\texttt{RGB-D}}$.

We optimize the following loss function over the parameters of networks $f_{\mathcal{D}}$, $h_{\mathcal{D}}$, $f_{\mathcal{C}}$, and g as:

$$\begin{aligned}&\mathcal{L}(\theta_D, \theta_h, \theta_C, \theta_g; \mathcal{T}^{\mathcal{S}}, \mathcal{T}^{\texttt{RGB-D}}) = \\ &\mathcal{L}_P(\theta_D, \theta_h; \mathcal{T}^{\mathcal{S}}) + \beta \mathcal{L}_{FM}(\theta_D, \theta_C, \theta_g; \mathcal{T}^{\texttt{RGB-D}}) + \gamma \mathcal{L}_{MMD}(\theta_D; \mathcal{T}^{\mathcal{S}}, \mathcal{T}^{\texttt{RGB-D}}) \,,\end{aligned} \tag{1}$$

where θ_D, θ_h, θ_C, and θ_g are the parameters of networks $f_{\mathcal{D}}$, $h_{\mathcal{D}}$, $f_{\mathcal{C}}$, and g, respectively. The losses $\mathcal{L}_P$ for the pose, $\mathcal{L}_{FM}$ for the feature mapping between color and depth features, and $\mathcal{L}_{MMD}$ for the MMD between synthetic and real depth images are weighted by meta parameters β and γ.
$\mathcal{L}_P$ is the sum of the errors for poses predicted from depth images:

$$\mathcal{L}_P(\theta_D, \theta_h; \mathcal{T}^{\mathcal{S}}) = \sum_{(\mathbf{x}_i^{\mathcal{S}}, \mathbf{y}_i) \in \mathcal{T}^{\mathcal{S}}} \| h_{\mathcal{D}}(f_{\mathcal{D}}(\mathbf{x}_i^{\mathcal{S}}; \theta_D); \theta_h) - \mathbf{y}_i \|^2 \,. \tag{2}$$

$\mathcal{L}_{FM}$ is the loss used to learn to map features extracted from depth images to features extracted from their corresponding color images:

$$\mathcal{L}_{FM}(\theta_D, \theta_C, \theta_g; \mathcal{T}^{\texttt{RGB-D}}) = \sum_{(\mathbf{x}_i^{\mathcal{R}}, \mathbf{x}_i^{\mathcal{D}}) \in \mathcal{T}^{\texttt{RGB-D}}} \| g(f_{\mathcal{C}}(\mathbf{x}_i^{\mathcal{R}}; \theta_C); \theta_g) - f_{\mathcal{D}}(\mathbf{x}_i^{\mathcal{D}}; \theta_D) \|^2 \,. \tag{3}$$

Finally, $\mathcal{L}_{MMD}$ is the Maximum Mean Discrepancy [17] loss to minimize the domain shift between the distribution of features extracted from real and synthetic depth images of these training sets:

$$\begin{aligned}\mathcal{L}_{MMD}(\theta_D; \mathcal{T}^{\mathcal{S}}, \mathcal{T}^{\texttt{RGB-D}}) = \Big\| &\tfrac{1}{|\mathcal{T}^{\texttt{RGB-D}}|} \textstyle\sum_{\mathbf{x}_i^{\mathcal{D}} \in \mathcal{T}^{\texttt{RGB-D}}} \phi(f_{\mathcal{D}}(\mathbf{x}_i^{\mathcal{D}}; \theta_D)) - \\ &\tfrac{1}{|\mathcal{T}^{\mathcal{S}}|} \textstyle\sum_{\mathbf{x}_i^{\mathcal{S}} \in \mathcal{T}^{\mathcal{S}}} \phi(f_{\mathcal{D}}(\mathbf{x}_i^{\mathcal{S}}; \theta_D)) \Big\|^2 \,,\end{aligned} \tag{4}$$

where $\phi(\cdot)$ denotes the mapping to kernel space, but the exact mapping is typically unknown in practice. By applying the kernel trick, this rewrites to:

$$\begin{aligned}\mathcal{L}_{MMD}(\theta_D; \mathcal{T}^{\mathcal{S}}, \mathcal{T}^{\texttt{RGB-D}}) = &\tfrac{1}{|\mathcal{T}^{\texttt{RGB-D}}|^2} \textstyle\sum_{i,i'} k(f_{\mathcal{D}}(\mathbf{x}_i^{\mathcal{D}}; \theta_D), f_{\mathcal{D}}(\mathbf{x}_{i'}^{\mathcal{D}}; \theta_D)) \\ &- \tfrac{2}{|\mathcal{T}^{\texttt{RGB-D}}||\mathcal{T}^{\mathcal{S}}|} \textstyle\sum_{i,j} k(f_{\mathcal{D}}(\mathbf{x}_i^{\mathcal{D}}; \theta_D), f_{\mathcal{D}}(\mathbf{x}_j^{\mathcal{S}}; \theta_D)) \\ &+ \tfrac{1}{|\mathcal{T}^{\mathcal{S}}|^2} \textstyle\sum_{j,j'} k(f_{\mathcal{D}}(\mathbf{x}_j^{\mathcal{S}}; \theta_D), f_{\mathcal{D}}(\mathbf{x}_{j'}^{\mathcal{S}}; \theta_D)) \,,\end{aligned} \tag{5}$$

where $k(\cdot, \cdot)$ denotes a kernel function. In this work, we implement $k(\cdot, \cdot)$ as an RBF kernel, such that

$$k(\mathbf{x}, \mathbf{y}) = \mathrm{e}^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} \,, \tag{6}$$

where we select the bandwidth $\sigma = 1$. Note that our method is not sensitive to the exact value of σ.

At run-time, given a real color image $\mathbf{x}^{\mathcal{R}}$, we extract its features in color space and map them to the depth feature space by the networks $f_{\mathcal{C}}$ and g, respectively, and then use the pose estimator $h_{\mathcal{D}}$ to predict the 3D pose $\hat{\mathbf{y}}$ of the object:

$$\hat{\mathbf{y}} = h_{\mathcal{D}}(g(f_{\mathcal{C}}(\mathbf{x}^{\mathcal{R}}))) \,. \tag{7}$$

3.2 Network Details and Optimization

3D Object Pose Estimation For the depth feature extraction network $f_{\mathcal{D}}$, we use a network architecture similar to the 50-layer Residual Network [37], and remove the Global Average Pooling [37] as done in [2,5,19]. The convolutional layers are followed by two fully-connected layers of 1024 neurons each. The pose estimator $h_{\mathcal{D}}$ consists of one single fully-connected layer with 16 outputs, which correspond to the 2D projections of the 8 corners of objects' 3D bounding box [4]. The 3D pose can then be computed from these 2D-3D correspondences with a PnP algorithm.

In order to do a fair comparison to [2,4] we use the same feature extractor, which consists of the first 10 pretrained convolutional layers of the VGG-16 network [38] and two fully-connected layers with 1024 neurons for the color feature extractor $f_{\mathcal{C}}$.

3D Hand Pose Estimation The architectures of $f_{\mathcal{D}}$ and $h_{\mathcal{D}}$ are the same as the ones used for 3D object pose estimation, except $h_{\mathcal{D}}$ outputs 3 values for each of the 21 joints, 63 in total. Additionally, we add a 3D pose prior [5] as a bottleneck layer to the pose estimator network $h_{\mathcal{D}}$, which was shown to efficiently constrain the 3D hand poses and also gives better performance in our case.

For the color feature extractor $f_{\mathcal{C}}$, we use the same architecture as the depth feature extractor, which makes the feature extractor comparable to the one used in [23,29].

Mapping Network and Optimization Following [2], we use a two Residual blocks [39] network g for mapping the features of size 1024 from color space to depth space. Each fully-connected layer within the Residual block has 1024 neurons.

In practice, we use $\beta = 0.02$ and $\gamma = 0.01$ for the meta parameters of the objective function in Eq. 1 for all our experiments. We first pretrain $f_{\mathcal{D}}$ and $h_{\mathcal{D}}$ on the synthetic depth dataset $\mathcal{T}^{\mathcal{S}}$. We also pretrain $f_{\mathcal{C}}$ by predicting depth from color images [40]. Pretraining is important in our experiments for improving convergence. We then jointly train all the networks together using the ADAM optimizer [41] with a batch size of 128 and an initial learning rate of 10^{-4}.

4 Evaluation

In this section, we evaluate our method on two different 3D pose estimation problems. We apply our method first on 3D rigid object pose estimation, and then on 3D hand pose estimation, both from color images only.

4.1 3D Object Pose Estimation from Color Images

We use the LINEMOD dataset [6] for benchmarking 3D object pose estimation. It consists of 13 texture-less objects, each registered with about 1200 real color

images and corresponding depth images under different viewpoints, and provided with the corresponding ground truth poses. For evaluating 3D object pose estimation methods using only color images, [1,2,4,42] use 15% of the images of the LINEMOD dataset for training and the rest for testing. This amounts to about 180 images per object for training, which had to be registered in 3D with the help of markers. In contrast to these methods, our approach does *not* require any labeled color image. Instead, it uses pairs of real images and depth images to learn mapping the features from color space to depth space.

Detection	Ground Truth Detection			Real Detection					
Metric	2D Projection [42]			2D Projection [42]			ADD [6]		
Method	BB8 [4]	Feature Mapping [2]	Ours	SSD-6D [25]	[26]	Ours	SSD-6D [25]	[26]	Ours
Ape	94.0	96.6	**97.3**	3.5	36.4	**96.9**	2.6	4.0	**19.8**
Bench Vise	90.0	**96.3**	92.7	0.9	30.5	**88.6**	15.1	20.9	**69.0**
Camera	81.7	**94.8**	83.4	1.0	56.0	**77.4**	6.1	30.5	**37.6**
Can	94.2	**96.6**	93.2	3.0	49.1	**91.3**	27.3	35.9	**42.3**
Cat	94.7	98.0	**98.7**	9.1	59.3	**98.0**	9.3	17.9	**35.4**
Driller	64.7	**83.3**	75.7	1.4	16.7	**72.2**	12.0	24.0	**54.7**
Duck	94.4	**96.3**	95.5	1.2	51.0	**91.8**	1.3	4.9	**29.4**
Egg Box	93.5	96.1	**97.1**	1.5	73.5	**92.0**	2.8	81.0	**85.2**
Glue	94.8	96.9	**97.3**	11.0	78.3	**92.4**	3.4	45.5	**77.8**
Hole Puncher	87.2	95.7	**97.2**	2.8	48.2	**96.8**	3.1	17.6	**36.0**
Iron	81.0	**92.3**	88.8	1.9	32.1	**85.9**	14.6	32.0	**63.1**
Lamp	76.2	83.5	**84.8**	0.5	30.8	**81.8**	11.4	60.5	**75.1**
Phone	70.6	88.2	**90.0**	5.3	53.3	**85.2**	9.7	33.8	**44.8**
Average	85.9	**93.4**	91.7	3.3	47.3	**88.5**	9.1	28.7	**51.6**

Table 1: Evaluation on the LINEMOD dataset [6]. The left part evaluates the impact of our proposed approach, where all methods predict the 3D object pose using the 2D projections of the objects' 3D bounding box [4], given the ground truth 2D object center without using pose refinement. Both BB8 [4] and Feature Mapping [2] use annotated color images, while our method achieves better performance than BB8 and similar performance to Feature Mapping without using any annotated color images at all. The middle and right parts show comparison of different pose estimation methods without using pose refinement, where no annotated color image is used for training. Our approach performs best.

In order to do a fair comparison and not learn any context of the scene, we extract the objects from both color images and depth images for generating the training set $\mathcal{T}^{\texttt{RGB-D}}$. We follow the protocol of [4] to augment the training data by rescaling the target object, adding a small pixel shift from the center of the image window and superimpose it on a random background. We pick random backgrounds from the RGB-D dataset [43] as they provide color images together with corresponding depth images.

For generating the training set $\mathcal{T}^{\mathcal{S}}$, given the CAD model of the target object, we randomly sample poses from the upper hemisphere of the object, within a

range of $[-45°, +45°]$ for the in-plane rotation and a camera distance within a range of $[65cm, 115cm]$. We also superimpose the rendered objects on a random depth background picked from the RGB-D dataset [43]. We apply a 5×5 median filter to mitigate the noise along the object boundaries. For both training sets $\mathcal{T}^{\texttt{RGB-D}}$ and $\mathcal{T}^{\mathcal{S}}$, we use image windows of size 128×128, and normalize them to the range of $[-1, +1]$.

To evaluate the impact of our approach, we first compare it to [4] and [2], which, similar to us, predict the 2D projection of the 3D object bounding box, followed by a PnP algorithm to estimate the 3D pose. The left part of Table 1 shows a comparison with these methods on the LINEMOD dataset by using 15% of real images for training and the ground truth 2D object center. We use the widely used 2D Projection metric [42] for comparison. We significantly outperform [4] and achieve similar performance to [2], which is the current state-of-the-art on the LINEMOD dataset. Most notably, we do not require any annotations for the color images. By using all the available pairs of real images and depth images, our approach performs with an accuracy of 95.7%. This shows that our approach almost eliminates the needs of the expensive task of annotating, simply by capturing data using an RGB-D camera.

We further compare to the approach of [25][3] without the extensive refinement step, which uses only synthetic color images for training. They obtain an accuracy of 3.3% on the 2D Projection metric. [4] trained only on synthetic color images also performs poorly with an accuracy of 12% on the same metric. This shows that while synthetic color images do not resemble real color images for training 3D pose estimation methods, our approach can effectively transfer features between color images and synthetic depth images. Although the domain randomization of [26] helps to increase the accuracy using synthetically generated images and generalize to different cameras, it is still not enough to bridge the domain gap. The comparisons are shown in Table 1.

Finally, we evaluate the domain adaption technique of [13] that aims to learn invariant features with respect to the shift between the color and depth domains. However, this performs with an accuracy of 2% using the 2D Projection metric, which shows that although this technique helps for general applications such as classification, the features are not well suited for 3D pose estimation.

4.2 3D Hand Pose Estimation from Color Images

We use the Stereo Hand Pose Benchmark (STB) [18] and the Rendered Hand Dataset (RHD) [22] for training and testing our approach for 3D hand pose estimation. The STB dataset contains 6 sequences each containing 1500 images of a stereo camera, and an RGB-D camera. It shows an user performing diverse hand gestures in front of different backgrounds. Each image is annotated with the corresponding 3D hand pose with 21 joints. The RHD dataset contains over 40k synthetically rendered images of humans performing various hand articulations.

[3] We used their public code to obtain accuracies on 2D Projection and ADD metrics.

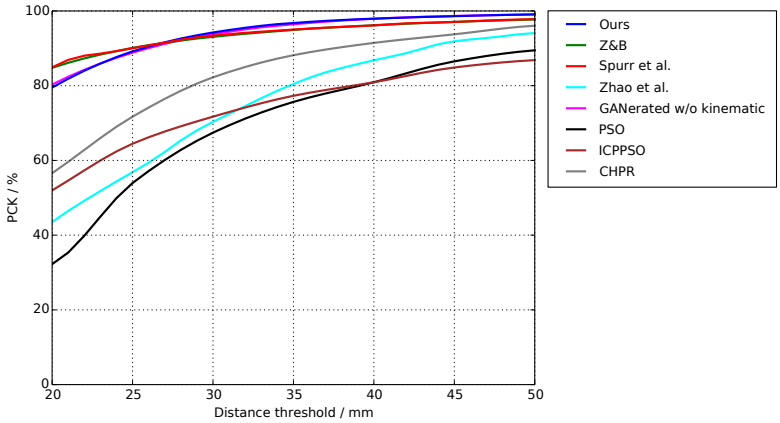


Fig. 4: 3D PCK curves for comparison to state-of-the-art 3D hand pose estimation methods on the STB dataset [18]. Note that all other approaches use annotated color images for training, whereas we do not use any annotations for the color images.

It consists of pairs of depth and color images, each with hand segmentation and 3D pose annotations of the same 21 joints.

We follow the protocol of [22,23,29], which use the first two sequences from the STB dataset for testing. The remaining ten sequences from the STB dataset together with the RHD dataset are used for the training set $\mathcal{T}^{\texttt{RGB-D}}$, since they both contain aligned depth and color images. Creating synthetic depth maps for hands is a relatively simple problem. For generating the training set $\mathcal{T}^{\mathcal{S}}$ we use the publicly available 3D hand model of [44] to render synthetic depth images of a hand. We use 5M synthetic images of the hand that are rendered online during training from poses of the NYU 3D hand pose dataset [44] perturbed with randomly added articulations. Furthermore, [22,23] align their 3D prediction to the ground truth wrist which we also do for comparison.

We use the pipeline provided by [5] to preprocess the depth images: It crops a 128×128 patch around the hand location, and normalizes its depth values to the range of $[-1, +1]$. For the color image we also crop a 128×128 patch around the corresponding hand location and subtract the mean RGB values. When a hand segmentation mask is available, such as for the RHD dataset [22], we superimpose the segmented hand on random color backgrounds from the RGB-D dataset [43]. During training, we randomly augment the hand scale, in-plane rotation, and 3D translation, as done in [5].

We compare to the following methods: GANerated [23][4], which uses a GAN to adapt synthetic color images for training a CNN; Z&B [22], which uses a

[4] The results reported in the paper [23] are tracking-based and include an additional inverse kinematics step. In order to make their results more comparable to ours, we denote results predicted for each frame separately without inverse kinematics kindly provided by the authors.

learned prior to lift 2D keypoints to 3D locations and the similar approach of Zhao *et al.* [45]; Zhang *et al.* [18], which use stereo images to estimate depth and apply a depth-based pose estimator with variants denoted PSO, ICPPSO, and CHPR; Spurr *et al.* [29], which project color images to a common embedding that is shared between images and 3D poses.

Fig. 4 shows the Percentage of Correct Keypoints (PCK) over different error thresholds, which is the most common metric on the STB dataset [18,22,23,29]. This metric denotes the average percentage of predicted joints below an Euclidean distance from the ground truth 3D joint location. While all methods that we compare to require annotations for color images, we can achieve comparable results without annotations of color images.

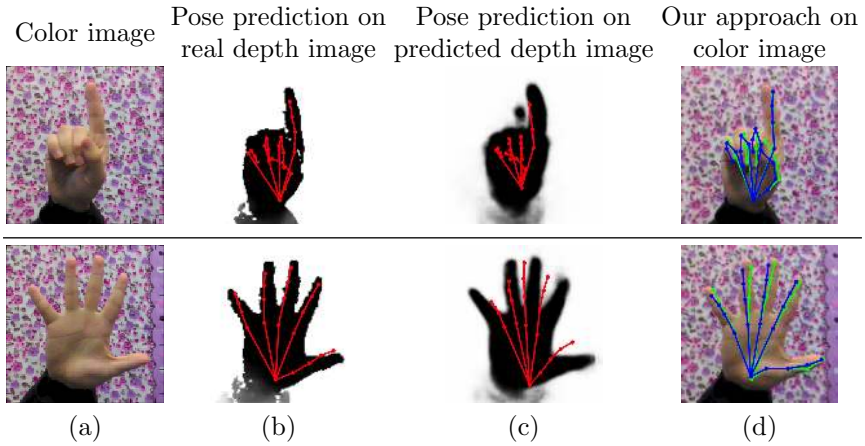


Fig. 5: We use the paired color and depth images shown in (a) and (b) to predict depth from color images [40] shown in (c). We further apply a 3D pose estimator [5] on these images. These predictions from the predicted and real depth images are shown in (b) and (c), respectively. Although the predicted depth images look visually close, the accuracy of the estimated 3D pose is significantly worse compared to using the real depth images. The results from our method are shown in (d) and provide a significantly higher accuracy. Our predictions are shown in blue and the ground truth in green.

3D hand pose estimation methods work very well on depth images [5,19]. Since we have paired color and depth images, we can train a CNN to predict the depth image from the corresponding color image [40]. Since the pose estimator works on cropped image patches, we only use these cropped image patches for depth prediction, which makes the task easier. We then use the predicted depth image for a depth-based 3D hand pose estimator with the pretrained model provided by the authors [5]. Although this approach also does not require any annotations of color images, our experiments show that this performs significantly worse on the STB dataset compared to ours. The 3D pose estimator gives

an average Euclidean joint error of 17.6mm on the real depth images and 39.8mm on the predicted depth images. We show a qualitative comparison in Fig. 5.

4.3 Qualitative Results

We show some qualitative results of our method for 3D object pose estimation and 3D hand pose estimation in Fig. 6. These examples show our approach predicts very close pose to the ground truth.

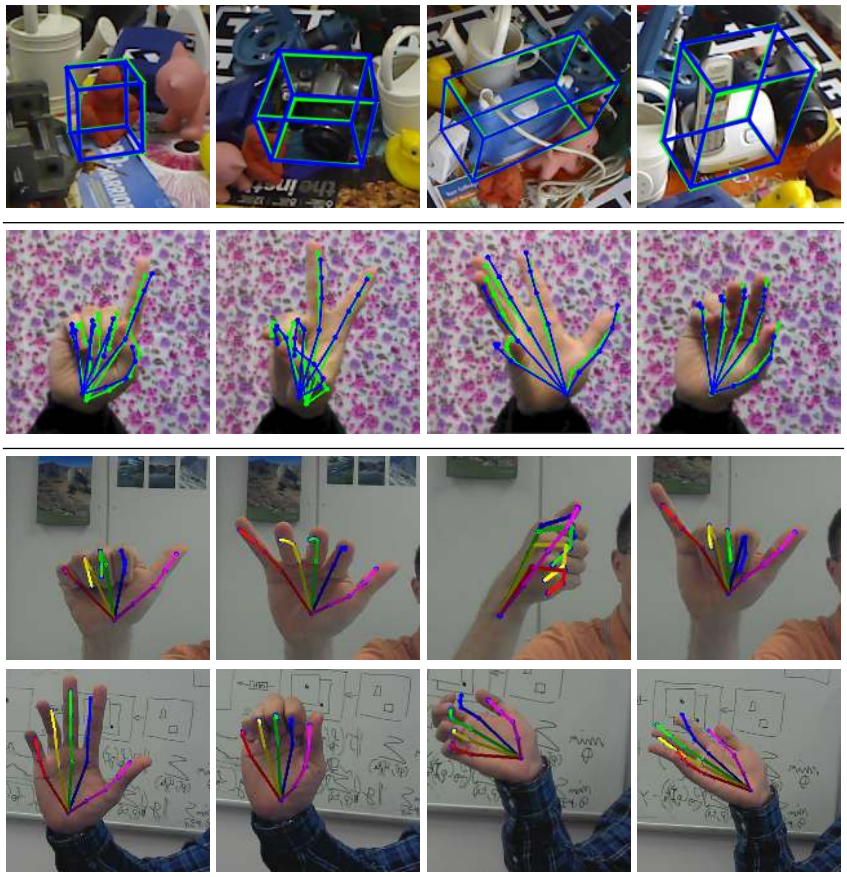
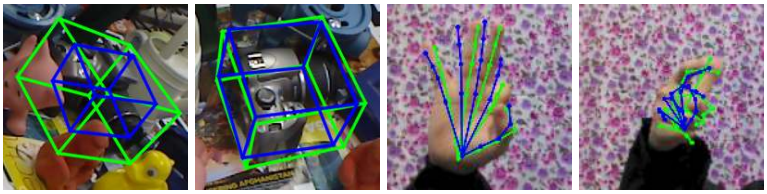


Fig. 6: Qualitative results of our method for 3D rigid object pose estimation on the LINEMOD dataset [6] (top row), and 3D hand pose estimation on the STB dataset [18] (middle row). Green denotes ground truth and blue corresponds to the predicted pose. We applied our trained network on real world RGB images of different users to estimate the 3D hand joint locations (bottom rows).

Fig. 7 illustrates some failure cases that occur due to the challenges of the test sets, such as partial occlusion that can easily be handled by training the



(a) (b) (c) (d)

Fig. 7: Some failure cases for 3D object pose estimation due to (a) partial occlusion, (b) not generalizing to every poses because of lack of corresponding color and depth images in the training set. Failure cases for 3D hand pose estimation due to (c) misalignment/confusion of the fingers, (d) severe self occlusion.

networks with partially occluded examples, or missing poses in the paired dataset $\mathcal{T}^{\texttt{RGB-D}}$ that can be simply resolved by capturing additional data with an RGB-D camera.

4.4 Computation Times

All experiments are implemented using Tensorflow and run on an Intel Core i7 3.3GHz desktop with a Geforce TITAN X. Given an image window extracted around the object, our approach takes 3.2ms for 3D object pose estimation to extract color features, map them to the depth feature space, and estimate the 3D pose. For 3D hand pose estimation, it takes 8.6ms. Training takes about 10 hours in our experiments.

5 Conclusion

In this work we presented a novel approach for 3D pose estimation from color images, without requiring labeled color images. We showed that a pose estimator can be trained on a large number of synthetic depth images, and at run-time, given a color image, we can map its features from color space to depth space. We showed that this mapping between the two domains can easily be learned by having corresponding color and depth images captured by a commodity RGB-D camera. Our approach is simple, general, and can be applied to different application domains, such as 3D rigid object pose estimation and 3D hand pose estimation. While for these tasks our approach achieves performances comparable to state-of-the-art methods, it does not require any annotations for the color images.

Acknowledgement: This work was supported by the Christian Doppler Laboratory for Semantic 3D Computer Vision, funded in part by Qualcomm Inc. We would like to thank Franziska Müller and Martin Sundermeyer for kindly providing additional evaluation results. Prof. V. Lepetit is a senior member of the *Institut Universitaire de France* (IUF).

References

1. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. RSS (2018)
2. Rad, M., Oberweger, M., Lepetit, V.: Feature Mapping for Learning Fast and Accurate 3D Pose Inference from Synthetic Images. In: CVPR. (2018)
3. Tekin, B., Sinha, S.N., Fua, P.: Real-Time Seamless Single Shot 6D Object Pose Prediction. In: CVPR. (2018)
4. Rad, M., Lepetit, V.: BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects Without Using Depth. In: ICCV. (2017)
5. Oberweger, M., Lepetit, V.: DeepPrior++: Improving Fast and Accurate 3D Hand Pose Estimation. In: ICCV Workshops. (2017)
6. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In: ACCV. (2012)
7. Hodan, T., Haluza, P., Obdrzalek, S., Matas, J., Lourakis, M., Zabulis, X.: T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects. In: WACV. (2017)
8. Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., Abbeel, P., Dollar, A.M.: Yale-CMU-Berkeley Dataset for Robotic Manipulation Research. IJRR **36** (2017) 261 – 268
9. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain Separation Networks. In: NIPS. (2016)
10. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Networks. In: NIPS. (2014)
11. Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., Webb, R.: Learning from Simulated and Unsupervised Images through Adversarial Training. In: CVPR. (2016)
12. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In: ICCV. (2017)
13. Ganin, Y., Lempitsky, V.: Unsupervised Domain Adaption by Backpropagation. In: ICML. (2015)
14. Muandet, K., Balduzzi, D., Schölkopf, B.: Domain Generalization via Invariant Feature Representation. In: ICML. (2013)
15. Pan, S., Tsang, I., Kwok, J., Yang, Q.: Domain Adaptation via Transfer Component Analysis. In: IJCAI. (2009)
16. Rozantsev, A., Salzmann, M., Fua, P.: Beyond Sharing Weights for Deep Domain Adaptation. In: CVPR. (2017)
17. Gretton, A., Borgwardt, K., Rasch, M.J., Schölkopf, B., Smola, A.J.: A Kernel Method for the Two-Sample Problem. In: NIPS. (2006)
18. Zhang, J., Jiao, J., Chen, M., Qu, L., Xu, X., Yang, Q.: 3D Hand Pose Tracking and Estimation Using Stereo Matching. ARXIV (2016)
19. Müller, F., Mehta, D., Sotnychenko, O., Sridhar, S., Casas, D., Theobalt, C.: Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor. In: ICCV. (2017)
20. Krull, A., Brachmann, E., Michel, F., Yang, M.Y., Gumhold, S., Rother, C.: Learning Analysis-By-Synthesis for 6D Pose Estimation in RGB-D Images. In: ICCV. (2015)
21. Oberweger, M., Rad, M., Lepetit, V.: Making Deep Heatmaps Robust to Partial Occlusions for 3D Object Pose Estimation. In: ECCV. (2018)

22. Zimmermann, C., Brox, T.: Learning to Estimate 3D Hand Pose from Single RGB Images. In: ICCV. (2017)
23. Müller, F., Bernard, F., Sotnychenko, O., Mehta, D., Sridhar, S., Casas, D., Theobalt, C.: GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB. In: CVPR. (2018)
24. Panteleris, P., Oikonomidis, I., Argyros, A.: Using a single RGB frame for real time 3D hand pose estimation in the wild. In: WACV. (2018)
25. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In: ICCV. (2017)
26. Sundermeyer, M., Marton, Z.C., Durner, M., Brucker, M., Triebel, R.: Implicit 3d orientation learning for 6d object detection from rgb images. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018)
27. Hinterstoisser, S., Lepetit, V., Wohlhart, P., Konolige, K.: On Pre-Trained Image Features and Synthetic Images for Deep Learning. In: ARXIV. (2017)
28. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. In: ICLR. (2014)
29. Spurr, A., Song, J., Park, S., Hilliges, O.: Cross-modal Deep Variational Hand Pose Estimation. In: CVPR. (2018)
30. Gupta, S., Hoffman, J., Malik, J.: Cross Modal Distillation for Supervision Transfer. In: CVPR. (2016)
31. Cai, G., Wang, Y., Zhou, M., He, L.: Unsupervised Domain Adaptation with Adversarial Residual Transform Networks. ARXIV (2018)
32. Zakharov, S., Planche, B., Wu, Z., Hutter, A., Kosch, H., Ilic, S.: Keep it Unreal: Bridging the Realism Gap for 2.5D Recognition with Geometry Priors Only. ARXIV (2018)
33. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In: IROS. (2017)
34. Hoffman, J., Gupta, S., Leong, J., Guadarrama, S., Darrell, T.: Cross-Modal Adaptation for RGB-D Detection. In: ICRA. (2016)
35. Song, X., Jiang, S., Herranz, L.: Combining Models from Multiple Sources for RGB-D Scene Recognition. In: IJCAI. (2017)
36. Huang, X., Peng, Y., Yuan, M.: Cross-modal Common Representation Learning by Hybrid Transfer Network. In: IJCAI. (2017)
37. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: CVPR. (2016)
38. Simonyan, K., Vedaldi, A., Zisserman, A.: Learning Local Feature Descriptors Using Convex Optimisation. PAMI (2014)
39. He, K., Zhang, X., Ren, R., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In: ICCV. (2015)
40. Eigen, D., Puhrsch, C., Fergus, R.: Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network. In: NIPS. (2014)
41. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: ICML. (2015)
42. Brachmann, E., Michel, F., Krull, A., Yang, M.M., Gumhold, S., Rother, C.: Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image. In: CVPR. (2016)
43. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: ICRA. (2011)
44. Tompson, J., Stein, M., LeCun, Y., Perlin, K.: Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks. TOG **33** (2014)
45. Zhao, R., Wang, Y., Martinez, A.: A Simple, Fast and Highly-Accurate Algorithm to Recover 3D Shape from 2D Landmarks on a Single Image. In: ARXIV. (2016)