Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation

Kiru Park, Timothy Patten and Markus Vincze
Vision for Robotics Laboratory, Automation and Control Institute, TU Wien, Austria
{park, patten, vincze}@acin.tuwien.ac.at

Abstract

Estimating the 6D pose of objects using only RGB images remains challenging because of problems such as occlusion and symmetries. It is also difficult to construct 3D models with precise texture without expert knowledge or specialized scanning devices. To address these problems, we propose a novel pose estimation method, Pix2Pose, that predicts the 3D coordinates of each object pixel without textured models. An auto-encoder architecture is designed to estimate the 3D coordinates and expected errors per pixel. These pixel-wise predictions are then used in multiple stages to form 2D-3D correspondences to directly compute poses with the PnP algorithm with RANSAC iterations. Our method is robust to occlusion by leveraging recent achievements in generative adversarial training to precisely recover occluded parts. Furthermore, a novel loss function, the transformer loss, is proposed to handle symmetric objects by guiding predictions to the closest symmetric pose. Evaluations on three different benchmark datasets containing symmetric and occluded objects show our method outperforms the state of the art using only RGB images.

1. Introduction

Pose estimation of objects is an important task to understand the given scene and operate objects properly in robotic or augmented reality applications. The inclusion of depth images has induced significant improvements by providing precise 3D pixel coordinates [10, 30]. However, depth images are not always easily available, e.g., mobile phones and tablets, typical for augmented reality applications, offer no depth data. As such, substantial research is dedicated to estimating poses of known objects using RGB images only.

A large body of work relies on the textured 3D model of an object, which is made by a 3D scanning device, e.g., BigBIRD Object Scanning Rig [35], and provided by a dataset to render synthetic images for training [14, 28] or refinement [18, 21]. Thus, the quality of texture in the 3D model should be sufficient to render visually correct images. Unfortunately, this is not applicable to domains that do not have textured 3D models such as industry that commonly use texture-less CAD models. Since the texture quality of a reconstructed 3D model varies with method, camera, and camera trajectory during the reconstruction process, it is difficult to guarantee sufficient quality for training. Therefore, it is beneficial to predict poses without textures on 3D models to achieve more robust estimation regardless of the texture quality.

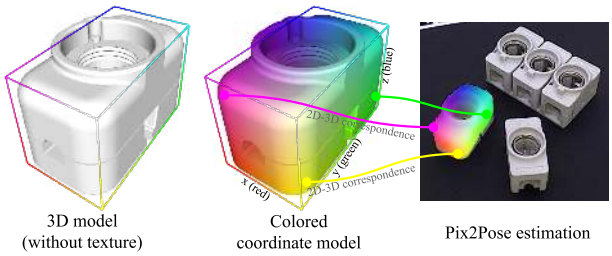

Figure 1. An example of converting a 3D model to a colored coordinate model. Normalized coordinates of each vertex are directly mapped to red, green and blue values in the color space. Pix2Pose predicts these colored images to build a 2D-3D correspondence per pixel directly without any feature matching operation.

Even though recent studies have shown great potential to estimate pose without textured 3D models using Convolutional Neural Networks (CNN) [2, 3, 24, 29], a significant challenge is to estimate correct poses when objects are occluded or symmetric. Training CNNs is often distracted by symmetric poses that have similar appearance inducing very large errors in a naïve loss function. In previous work, a strategy to deal with symmetric objects is to limit the range of poses while rendering images for training [14, 22] or simply to apply a transformation from the pose outside of the limited range to a symmetric pose within the range [24] for real images with pose annotations. This approach is sufficient for objects that have infinite and continuous symmetric poses on a single axis, such as cylinders, by simply ignoring the rotation about the axis. However, as pointed in [24], when an object has a finite number of symmetric poses, it is difficult to determine poses around the boundaries of view limits. For example, if a box has an angle of symmetry, π,

with respect to an axis and a view limit between 0 and π, the pose at $\pi + \alpha(\alpha \approx 0, \alpha > 0)$ has to be transformed to a symmetric pose at α even if the detailed appearance is closer to a pose at π. Thus, a loss function has to be investigated to guide pose estimations to the closest symmetric pose instead of explicitly defined view ranges.

This paper proposes a novel method, *Pix2Pose*, that can supplement any 2D detection pipeline for additional pose estimation. Pix2Pose predicts pixel-wise 3D coordinates of an object using RGB images without textured 3D models for training. The 3D coordinates of occluded pixels are implicitly estimated by the network in order to be robust to occlusion. A specialized loss function, the *transformer loss*, is proposed to robustly train the network with symmetric objects. As a result of the prediction, each pixel forms a 2D-3D correspondence that is used to compute poses by the Perspective-n-Point algorithm (PnP) [17].

To summarize, the contributions of the paper are: **(1)** A novel framework for 6D pose estimation, Pix2Pose, that robustly regresses pixel-wise 3D coordinates of objects from RGB images using 3D models without textures during training. **(2)** A novel loss function, the transformer loss, for handling symmetric objects that have a finite number of ambiguous views. **(3)** Experimental results on three different datasets, LineMOD [8], LineMOD Occlusion [1], and T-Less [9], showing that Pix2Pose outperforms the state-of-the-art methods even if objects are occluded or symmetric.

The remainder of this paper is organized as follows. A brief summary of related work is provided in Sec. 2. Details of Pix2Pose and the pose prediction process are explained in Sec. 3 and Sec. 4. Experimental results are reported in Sec. 5 to compare our approach with the state-of-the-art methods. The paper concludes in Sec. 6.

2. Related work

This section gives a brief summary of previous work related to pose estimation using RGB images. Three different approaches for pose estimation using CNNs are discussed and the recent advances of generative models are reviewed.

CNN based pose estimation The first, and simplest, method to estimate the pose of an object using a CNN is to predict a representation of a pose directly such as the locations of projected points of 3D bounding boxes [24, 29], classified view points [14], unit quaternions and translations [32], or the Lie algebra representation, $so(3)$, with the translation of z-axis [3]. Except for methods that predict projected points of the 3D bounding box, which requires further computations for the PnP algorithm, the direct regression is computationally efficient since it does not require additional computation for the pose. The drawback of these methods, however, is the lack of correspondences that can be useful to generate multiple pose hypotheses for the robust estimation of occluded objects. Furthermore, symmetric objects are usually handled by limiting the range of viewpoints, which sometimes requires additional treatments, e.g., training a CNN for classifying view ranges [24]. Xiang *et al*. [32] propose a loss function that computes the average distance to the nearest points of transformed models in an estimated pose and an annotated pose. However, searching for the nearest 3D points is time consuming and makes the training process inefficient.

The second method is to match features to find the nearest pose template and use the pose information of the template as an initial guess [8]. Recently, Sundermeyer *et al*. [28] propose an auto-encoder network to train implicit representations of poses without supervision using RGB images only. Manual handling of symmetric objects is not necessary for this work since the implicit representation can be close to any symmetric view. However, it is difficult to specify 3D translations using rendered templates that only give a good estimation of rotations. The size of the 2D bounding box is used to compute the z-component of 3D translation, which is too sensitive to small errors of 2D bounding boxes that are given from a 2D detection method.

The last method is to predict 3D locations of pixels or local shapes in the object space [2, 15, 22]. Brachmann *et al*. [2] regress 3D coordinates and predict a class for each pixel using the auto-context random forest. Oberwerger *et al*. [22] predict multiple heat-maps to localize the 2D projections of 3D points of objects using local patches. These methods are robust to occlusion because they focus on local information only. However, additional computation is required to derive the best result among pose hypotheses, which makes these methods slow.

The method proposed in this paper belongs to the last category that predicts 3D locations of pixels in the object frame as in [1, 2]. Instead of detecting an object using local patches from sliding windows, an independent 2D detection network is employed to provide areas of interest for target objects as performed in [28].

Generative models Generative models using auto-encoders have been used to de-noise [31] or recover the missing parts of images [33]. Recently, using Generative Adversarial Network (GAN) [5] improves the quality of generated images that are less blurry and more realistic, which are used for the image-to-image translation [13], image in-painting and de-noising [11, 23] tasks. Zakharov *et al*. [34] propose a GAN based framework to convert a real depth image to a synthetic depth image without noise and background for classification and pose estimation.

Inspired by previous work, we train an auto-encoder architecture with GAN to convert color images to coordinate values accurately as in the image-to-image translation task while recovering values of occluded parts as in the image in-painting task.

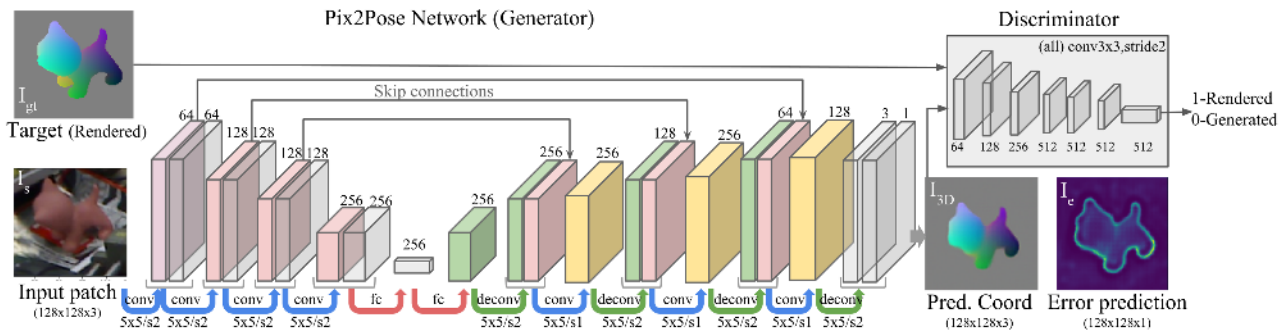

Figure 2. An overview of the architecture of Pix2Pose and the training pipeline.

3. Pix2Pose

This section provides a detailed description of the network architecture of Pix2Pose and loss functions for training. As shown in Fig. 2, Pix2Pose predicts 3D coordinates of individual pixels using a cropped region containing an object. The robust estimation is established by recovering 3D coordinates of occluded parts and using all pixels of an object for pose prediction. A single network is trained and used for each object class. The texture of a 3D model is not necessary for training and inference.

3.1. Network Architecture

The architecture of the Pix2Pose network is described in Fig. 2. The input of the network is a cropped image I_{s} using a bounding box of a detected object class. The outputs of the network are normalized 3D coordinates of each pixel I_{3D} in the object coordinate and estimated errors I_{e} of each prediction, $I_{\mathrm{3D}}, I_{\mathrm{e}} = G(I_{\mathrm{s}})$, where G denotes the Pix2Pose network. The target output includes coordinate predictions of occluded parts, which makes the prediction more robust to partial occlusion. Since a coordinate consists of three values similar to RGB values in an image, the output I_{3D} can be regarded as a color image. Therefore, the ground truth output is easily derived by rendering the colored coordinate model in the ground truth pose. An example of 3D coordinate values in a color image is visualized in Fig. 1. The error prediction I_e is regarded as a confidence score of each pixel, which is directly used to determine outlier and inlier pixels before the pose computation.

The cropped image patch is resized to $128\times128px$ with three channels for RGB values. The sizes of filters and channels in the first four convolutional layers, the encoder, are the same as in [28]. To maintain details of low-level feature maps, skip connections [27] are added by copying the half channels of outputs from the first three layers to the corresponding symmetric layers in the decoder, which results in more precise estimation of pixels around geometrical boundaries. The filter size of every convolution and deconvolution layer is fixed to 5×5 with stride 1 or 2 denoted as *s1* or *s2* in Fig. 2. Two fully connected layers are applied for the bottle neck with 256 dimensions between the encoder and the decoder. The batch normalization [12] and the *LeakyReLU* activation are applied to every output of the intermediate layers except the last layer. In the last layer, an output with three channels and the $\tanh$ activation produces a 3D coordinate image I_{3D}, and another output with one channel and the $\mathrm{sigmoid}$ activation estimates the expected errors I_{e}.

3.2. Network Training

The main objective of training is to predict an output that minimizes errors between a target coordinate image and a predicted image while estimating expected errors of each pixel.

Transformer loss for 3D coordinate regression To reconstruct the desired target image, the average L1 distance of each pixel is used. Since pixels belonging to an object are more important than the background, the errors under the object mask are multiplied by a factor of β (≥ 1) to weight errors in the object mask. The basic reconstruction loss $\mathcal{L}_{\mathrm{r}}$ is defined as,

$$\mathcal{L}_{\mathrm{r}} = \frac{1}{n}\Big[\beta \sum_{i\in M} ||I^i_{\mathrm{3D}} - I^i_{\mathrm{gt}}||_1 + \sum_{i\notin M} ||I^i_{\mathrm{3D}} - I^i_{\mathrm{gt}}||_1\Big], \quad (1)$$

where n is the number of pixels, I^i_{gt} is the i^{th} pixel of the target image, and M denotes an object mask of the target image, which includes pixels belonging to the object when it is fully visible. Therefore, this mask also contains the occluded parts to predict the values of invisible parts for robust estimation of occluded objects.

The loss above cannot handle symmetric objects since it penalizes pixels that have larger distances in the 3D space without any knowledge of the symmetry. Having the advantage of predicting pixel-wise coordinates, the 3D coordinate of each pixel is easily transformed to a symmetric pose by multiplying a 3D transformation matrix to the target image directly. Hence, the loss can be calculated for a pose that

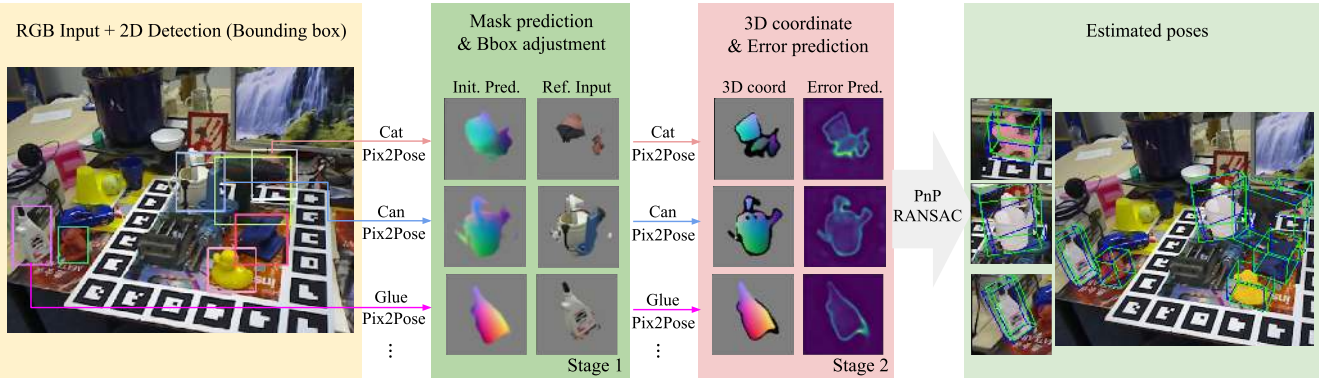

Figure 3. An example of the pose estimation process. An image and 2D detection results are the input. In the first stage, the predicted results are used to specify important pixels and adjust bounding boxes while removing backgrounds and uncertain pixels. In the second stage, pixels with valid coordinate values and small error predictions are used to estimate poses using the PnP algorithm with RANSAC. Green and blue lines in the result represent 3D bounding boxes of objects in ground truth poses and estimated poses.

has the smallest error among symmetric pose candidates as formulated by,

$$\mathcal{L}_{\text{3D}} = \min_{p \in \text{sym}} \mathcal{L}_{\text{r}}(I_{\text{3D}}, R_p I_{gt}), \tag{2}$$

where $R_p \in \mathbb{R}^{3\text{x}3}$ is a transformation from a pose to a symmetric pose in a pool of symmetric poses, sym, including an identity matrix for the given pose. The pool sym is assumed to be defined before the training of an object. This novel loss, the transformer loss, is applicable to any symmetric object that has a finite number of symmetric poses. This loss adds only a tiny effort for computation since a small number of matrix multiplications is required. The transformer loss in Eq. 2 is applied instead of the basic reconstruction loss in Eq. 1. The benefit of the transformer loss is analyzed in Sec. 5.7.

Loss for error prediction The error prediction I_e estimates the difference between the predicted image I_{3D} and the target image I_{gt}. This is identical to the reconstruction loss $\mathcal{L}_{\text{r}}$ with $\beta = 1$ such that pixels under the object mask are not penalized. Thus, the error prediction loss $\mathcal{L}_{\text{e}}$ is written as,

$$\mathcal{L}_{\text{e}} = \frac{1}{n} \sum_i ||I^i_{\text{e}} - \min\left[\mathcal{L}^i_{\text{r}}, 1\right]||^2_2, \beta = 1. \tag{3}$$

The error is bounded to the maximum value of the sigmoid function.

Traininig with GAN As discussed in Sec. 2, the network training with GAN generates more precise and realistic images in a target domain using images of another domain [13]. The task for Pix2Pose is similar to this task since it converts a color image to a 3D coordinate image of an object. Therefore, the discriminator and the loss function of GAN [5], $\mathcal{L}_{\text{GAN}}$, is employed to train the network. As shown in Fig. 2, the discriminator network attempts to distinguish whether the 3D coordinate image is rendered by a 3D model or is estimated. The loss is defined as,

$$\mathcal{L}_{\text{GAN}} = \log D(I_{gt}) + \log(1 - D(G(I_{\text{src}}))), \tag{4}$$

where D denotes the discriminator network. Finally, the objective of the training with GAN is formulated as,

$$G^* = \arg\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) + \lambda_1 \mathcal{L}_{\text{3D}}(G) + \lambda_2 \mathcal{L}_{\text{e}}(G), \tag{5}$$

where λ_1 and λ_2 denote weights to balance different tasks.

4. Pose prediction

This section gives a description of the process that computes a pose using the output of the Pix2Pose network. The overview of the process is shown in Fig. 3. Before the estimation, the center, width, and height of each bounding box are used to crop the region of interest and resize it to the input size, $128 \times 128px$. The width and height of the region are set to the same size to keep the aspect ratio by taking the larger value. Then, they are multiplied by a factor of 1.5 so that the cropped region potentially includes occluded parts. The pose prediction is performed in two stages and the identical network is used in both stages. The first stage aligns the input bounding box to the center of the object which could be shifted due to different 2D detection methods. It also removes unnecessary pixels (background and uncertain) that are not preferred by the network. The second stage predicts a final estimation using the refined input from the first stage and computes the final pose.

Stage 1: Mask prediction and Bbox Adjustment In this stage, the predicted coordinate image I_{3D} is used for specifying pixels that belong to the object including the occluded

parts by taking pixels with non-zero values. The error prediction is used to remove the uncertain pixels if an error for a pixel is larger than the outlier threshold θ_o. The valid object mask is computed by taking the union of pixels that have non-zero values and pixels that have lower errors than θ_o. The new center of the bounding box is determined with the centroid of the valid mask. As a result, the output of the first stage is a refined input that only contains pixels in the valid mask cropped from a new bounding box. Examples of outputs of the first stage are shown in Fig. 3. The refined input possibly contains the occluded parts when the error prediction is below the outlier threshold θ_o, which means the coordinates of these pixels are easy to predict despite occlusions.

Stage 2: Pixel-wise 3D coordinate regression with errors The second estimation with the network is performed to predict a coordinate image and expected error values using the refined input as depicted in Fig. 3. Black pixels in the 3D coordinate samples denote points that are removed when the error prediction is larger than the inlier threshold θ_i even though points have non-zero coordinate values. In other words, pixels that have non-zero coordinate values with smaller error predictions than θ_i are used to build 2D-3D correspondences. Since each pixel already has a value for a 3D point in the object coordinate, the 2D image coordinates and predicted 3D coordinates directly form correspondences. Then, applying the PnP algorithm [17] with RANdom SAmple Consensus (RANSAC) [4] iteration computes the final pose by maximizing the number of inliers that have lower re-projection errors than a threshold θ_{re}. It is worth mentioning that there is no rendering involved during the pose estimation since Pix2Pose does not assume textured 3D models. This also makes the estimation process fast.

5. Evaluation

In this section, experiments on three different datasets are performed to compare the performance of Pix2Pose to state-of-the-art methods. The evaluation using LineMOD [8] shows the performance for objects without occlusion in the single object scenario. For the multiple object scenario with occlusions, LineMOD Occlusion [1] and T-Less [9] are used. The evaluation on T-Less shows the most significant benefit of Pix2Pose since T-Less provides texture-less CAD models and most of the objects are symmetric, which is more challenging and common in industrial domains.

5.1. Augmentation of training data

A small number of real images are used for training with various augmentations. Image pixels of objects are extracted from real images and pasted to background images that are randomly picked from the Coco dataset [20]. After applying the color augmentations on the image, the borderlines between the object and the background are blurred to make smooth boundaries. A part of the object area is replaced by the background image to simulate occlusion. Lastly, a random rotation is applied to both the augmented color image and the target coordinate image. The same augmentation is applied to all evaluations except sizes of occluded areas that need to be larger for datasets with occlusions, LineMOD Occlusion and T-Less. Sample augmentated images are shown in Fig. 4. As explained in Sec. 4, the network recognizes two types of inputs, with background in the first stage and without background pixels in the second stage. Thus, a mini-batch is altered for every iteration as shown in Fig. 4. Target coordinate images are rendered before training by placing the object in the ground truth poses using the colored coordinate model as in Fig. 1.

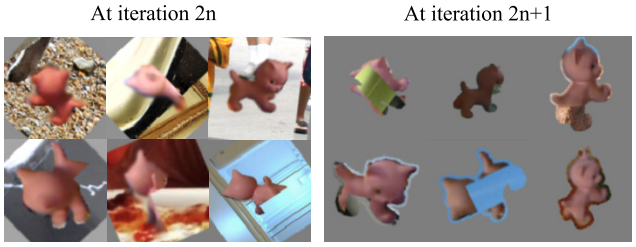

Figure 4. Examples of mini-batches for training. A mini-batch is altered for every training iteration. Left: images for the first stage, Right: images for the second stage.

5.2. Implementation details

For training, the batch size of each iteration is set to 50, the Adam optimizer [16] is used with initial learning rate of 0.0001 for 25K iterations. The learning rate is multiplied by a factor of 0.1 for every 12K iterations. Weights of loss functions in Eq. 1 and Eq. 5 are: β=3, λ_1=100 and λ_2=50. For evaluation, a 2D detection network and Pix2Pose networks of all object candidates in test sequences are loaded to the GPU memory, which requires approximately 2.2GB for the LineMOD Occlusion experiment with eight objects. The standard parameters for the inference are: θ_i=0.1, θ_o=[0.1, 0.2, 0.3], and θ_{re}=3. Since the values of error predictions are biased by the level of occlusion in the online augmentation and the shape and size of each object, the outlier threshold θ_o in the first stage is determined among three values to include more numbers of visible pixels while excluding noisy pixels using samples of training images with artificial occlusions. More details about parameters are given in the supplementary material. The training and evaluations are performed with an Nvidia GTX 1080 GPU and i7-6700K CPU.

2D detection network An improved Faster R-CNN [6, 26] with Resnet-101 [7] and Retinanet [19] with Resnet-50 are employed to provide classes of detected objects with

	ape	bvise	cam	can	cat	driller	duck	e.box*	glue*	holep	iron	lamp	phone	avg
Pix2Pose	**58.1**	**91.0**	**60.9**	**84.4**	**65.0**	**76.3**	**43.8**	**96.8**	79.4	**74.8**	**83.4**	**82.0**	45.0	**72.4**
Tekin [29]	21.6	81.8	36.6	68.8	41.8	63.5	27.2	69.6	**80.0**	42.6	75.0	71.1	**47.7**	56.0
Brachmann [2]	33.2	64.8	38.4	62.9	42.7	61.9	30.2	49.9	31.2	52.8	80.0	67.0	38.1	50.2
BB8 [24]	27.9	62.0	40.1	48.1	45.2	58.6	32.8	40.0	27.0	42.4	67.0	39.9	35.2	43.6
Lienet$^{30\%}$ [3]	38.8	71.2	52.5	86.1	66.2	82.3	32.5	79.4	63.7	56.4	65.1	89.4	65.0	65.2
BB8^{ref} [24]	40.4	91.8	55.7	64.1	62.6	74.4	44.3	57.8	41.2	67.2	84.7	76.5	54.0	62.7
Implicit^{syn} [28]	4.0	20.9	30.5	35.9	17.9	24.0	4.9	81.0	45.5	17.6	32.0	60.5	33.8	31.4
SSD-6D$^{syn/ref}$ [14]	65	80	78	86	70	73	66	100	100	49	78	73	79	76.7
Rad$^{syn/ref}$ [25]	-	-	-	-	-	-	-	-	-	-	-	-	-	***78.7***

Table 1. LineMOD: Percentages of correctly estimated poses (AD{D|I}-10%). ($^{30\%}$) means the training images are obtained from 30% of test sequences that are two times larger than ours. (^{ref}) denotes the results are derived after iterative refinement using textured 3D models for rendering. (^{syn}) indicates the method uses synthetically rendered images for training that also needs textured 3D models.

2D bounding boxes for all target objects of each evaluation. The networks are initialized with pre-trained weights using the Coco dataset [20]. The same set of real training images is used to generate training images. Cropped patches of objects in real images are pasted to random background images to generate training images that contain multiple classes in each image.

5.3. Metrics

A standard metric for LineMOD, AD{D|I}, is mainly used for the evaluation [8]. This measures the average distance of vertices between a ground truth pose and an estimated pose. For symmetric objects, the average distance to the nearest vertices is used instead. The pose is considered correct when the error is less than 10% of the maximum 3D diameter of an object.

For T-Less, the Visible Surface Discrepancy (VSD) is used as a metric since the metric is employed to benchmark various 6D pose estimation methods in [10]. This metric measures distance errors of visible parts only, which makes the metric invariant to ambiguities caused by symmetries and occlusion. As in previous work, the pose is regarded as correct when the error is less than 0.3 with τ=20mm and δ=15mm.

5.4. LineMOD

For training, test sequences are separated into a training and test set. The divided set of each sequence is identical to the work of [2, 29], which uses 15% of test scenes, approximately less than 200 images per object, for training. A detection result, using Faster R-CNN, of an object with the highest score in each scene is used for pose estimation since the detection network produces multiple results for all 13 objects. For the symmetric objects, marked with (*) in Table 1, the pool of symmetric poses sym is defined as, sym= $[I, R_z^{\pi}]$, where R_z^{π} represents a transformation matrix of rotation with π about the z-axis.

The upper part of Table 1 shows Pix2Pose significantly outperforms state-of-the-art methods that use the same amount of real training images without textured 3D models. Even though methods on the bottom of Table 1 use a larger portion of training images, use textured 3D models for training or pose refinement, our method shows competitive results against these methods. The results on symmetric objects show the best performance among methods that do not perform pose refinement. This verifies the benefit of the transformer loss, which improves the robustness of initial pose predictions for symmetric objects.

Method	**Pix2Pose**	Oberweger† [22]	PoseCNN† [32]	Tekin [29]
ape	**22.0**	17.6	9.6	2.48
can	44.7	**53.9**	45.2	17.48
cat	**22.7**	3.31	0.93	0.67
driller	44.7	**62.4**	41.4	7.66
duck	15.0	19.2	**19.6**	1.14
eggbox*	25.2	**25.9**	22.0	-
glue*	32.4	**39.6**	38.5	10.08
holep	**49.5**	21.3	22.1	5.45
Avg	**32.0**	30.4	24.9	6.42

Table 2. LineMOD Occlusion: object recall (AD{D|I}-10%). (†) indicates the method uses synthetically rendered images and real images for training, which has better coverage of viewpoints.

5.5. LineMOD Occlusion

LineMOD Occlusion is created by annotating eight objects in a test sequence of LineMOD. Thus, the test sequences of eight objects in LineMOD are used for training without overlapping with test images. Faster R-CNN is used as a 2D detection pipeline.

As shown in Table 2, Pix2Pose significantly outperforms the method of [29] using only real images for training. Furthermore, Pix2Pose outperforms the state of the art on three out of eight objects. On average it performs best even

Input	RGB only		RGB-D	
Method	**Pix2Pose**	Implicit [28]	Kehl [15]	Brachmann [2]
Avg	**29.5**	18.4	24.6	17.8

Table 3. T-Less: object recall ($e_{\text{VSD}} < 0.3$, τ = 20mm) on all test scenes using PrimeSense. Results of [15] and [2] are cited from [10]. Object-wise results are included in the supplement material.

though methods of [22] and [32] use more images that are synthetically rendered by using textured 3D models of objects. Although these methods cover more various poses than the given small number of images, Pix2Pose robustly estimates poses with less coverage of training poses.

5.6. T-Less

In this dataset, a CAD model without textures and a reconstructed 3D model with textures are given for each object. Even though previous work uses reconstructed models for training, to show the advantage of our method, CAD models are used for training (as shown in Fig. 1) with real training images provided by the dataset. To minimize the gap of object masks between a real image and a rendered scene using a CAD model, the object mask of the real image is used to remove pixels outside of the mask in the rendered coordinate images. The pool of symmetric poses sym of objects is defined manually similar to the *eggbox* in the LineMOD evaluation for box-like objects such as *obj-05*. For cylindrical objects such as *obj-01*, the rotation component of the z-axis is simply ignored and regarded as a non-symmetric object. The experiment is performed based on the protocol of [10]. Instead of a subset of the test sequences in [10], full test images are used to compare with the state of the art [28]. Retinanet is used as a 2D detection method and objects visible more than 10% are considered as estimation targets [10, 28].

The result in Table 3 shows Pix2Pose outperforms the-state-of-the-art method that uses RGB images only by a significant margin. The performance is also better than the best learning-based methods [2, 15] in the benchmark [10]. Although these methods use color and depth images to refine poses or to derive the best pose among multiple hypotheses, our method, that predicts a single pose per detected object, performs better than these methods without refinement using depth images.

5.7. Ablation studies

In this section, we present ablation studies by answering four important questions that clarify the contribution of each component in the proposed method.

How does the transformer loss perform? The *obj-05* in T-Less is used to analyze the variation of loss values with respect to symmetric poses and to show the contribution of

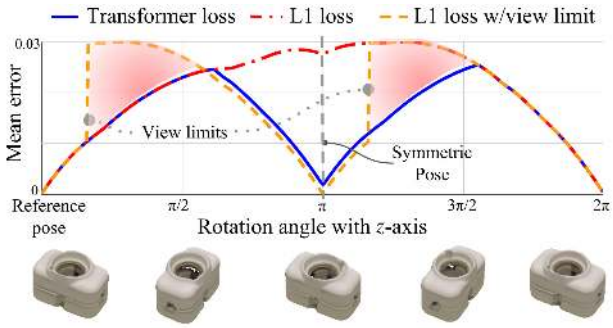

Figure 5. Variation of the reconstruction loss for a symmetric object with respect to z-axis rotation using obj-05 in T-Less [9].

Transformer loss	L1-view limits	L1
55.2	47.2	33.4

Table 4. Recall ($e_{\text{vsd}} < 0.3$) of obj-05 in T-Less using different reconstruction losses for training.

the transformer loss. To see the variation of loss values, 3D coordinate images are rendered while rotating the object around the z-axis. Loss values are computed using the coordinate image of a reference pose as a target output I_{gt} and images of other poses as predicted outputs I_{3D} in Eq. 1 and Eq. 2. As shown in Fig. 5, the L1 loss in Eq. 1 produces large errors for symmetric poses around π, which is the reason why the handling of symmetric objects is required. On the other hand, the value of the transformer loss produces minimum values on 0 and π, which is expected for *obj-05* with an angle of symmetry of π. The result denoted by *view limits* shows the value of the L1 loss while limiting the z-component of rotations between 0 and π. The pose that exceeds this limit is rotated to a symmetric pose. As discussed in Sec. 1, values are significantly changed at the angles of view limits and over-penalize poses under areas with red in Fig. 5, which causes noisy predictions of poses around these angles. The results in Table 4 show the transformer loss significantly improves the performance compared to the L1 loss with the view limiting strategy and the L1 loss without handling symmetries.

What if the 3D model is not precise? The evaluation on T-Less already shows the robustness to 3D CAD models that have small geometric differences with real objects. However, it is often difficult to build a 3D model or a CAD model with refined meshes and precise geometries of a target object. Thus, a simpler 3D model, a convex hull covering out-bounds of the object, is used in this experiment as shown in Fig. 6. The training and evaluation are performed in the same way for the LineMOD evaluation with synchronization of object masks using annotated masks of real images. As shown in the top-left of Fig. 6, the performance slightly drops when using the convex hull. However, the performance is still competitive with methods that use

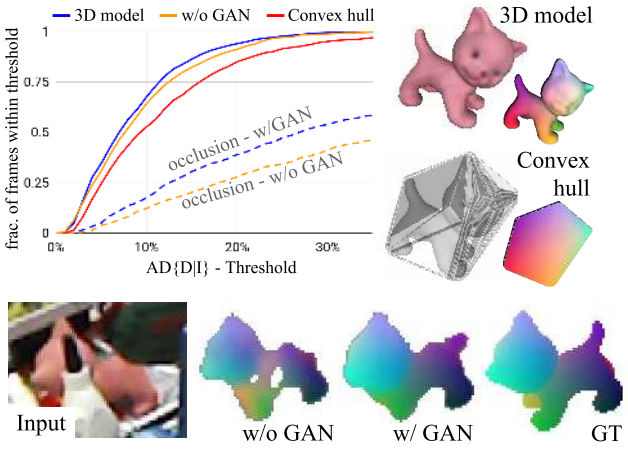

Figure 6. Top: the fraction of frames within AD{D|I} thresholds for the *cat* in LineMOD. The larger area under a curve means better performance. Bottom: qualitative results with/without GAN.

3D bounding boxes of objects, which means that Pix2Pose uses the details of 3D coordinates for robust estimation even though 3D models are roughly reconstructed.

Does GAN improve results? The network of Pix2Pose can be trained without GAN by removing the GAN loss in the final loss function in Eq. 5. Thus, the network only attempts to reconstruct the target image without trying to trick the discriminator. To compare the performance, the same training procedure is performed without GAN until the loss value excluding the GAN loss reaches the same level. Results in the top-left in Fig. 6 shows the fraction of correctly estimated poses with varied thresholds for the ADD metric. Solid lines show the performance on the original LineMOD test images, which contains fully visible objects, and dashed lines represent the performance on the same test images with artificial occlusions that are made by replacing 50% of areas in each bounding box with zero. There is no significant change in the performance when objects are fully visible. However, the performance drops significantly without GAN when objects are occluded. Examples in the bottom of Fig. 6 also show training with GAN produces robust predictions on occluded parts.

Is Pix2Pose robust to different 2D detection networks? Table 5 reports the results using different 2D detection networks on LineMOD. Retinanet and Faster R-CNN are trained using the same training images used in the LineMOD evaluation. In addition, the public code and trained weights of SSD-6D [14] are used to derive 2D detection results while ignoring pose predictions of the network. It is obvious that pose estimation results are proportional to 2D detection performances. On the other hand, the portion of correct poses on good bounding boxes (those that overlap more than 50% with ground truth) does not change significantly. This shows that Pix2Pose is robust to different 2D detection results when a bounding box overlaps the target object sufficiently. This robustness is accomplished by the refinement in the first stage that extracts useful pixels with a re-centered bounding box from a test image. Without the two stage approach, the performance significantly drops to 41% on LineMOD when the output of the network in the first stage is used directly for the PnP computation.

	SSD-6D [14]	Retina [19]	R-CNN [26]	GT bbox
2D bbox	89.1	97.7	98.6	100
6D pose	64.0	71.1	72.4	74.7
6D pose/2D bbox	70.9	72.4	73.2	74.7

Table 5. Average percentages of correct 2D bounding boxes (IoU>0.5) and correct 6D poses (ADD-10%) on LineMOD using different 2D detection methods. The last row reports the percentage of correctly estimated poses on scenes that have correct bounding boxes (IoU>0.5).

5.8. Inference time

The inference time varies according to the 2D detection networks. Faster R-CNN takes 127ms and Retinanet takes 76ms to detect objects from an image with $640\times480px$. The pose estimation for each bounding box takes approximately 25-45ms per region. Thus, our method is able to estimate poses at 8-10 fps with Retinanet and 6-7 fps with Faster R-CNN in the single object scenario.

6. Conclusion

This paper presented a novel architecture, Pix2Pose, for 6D object pose estimation from RGB images. Pix2Pose addresses several practical problems that arise during pose estimation: the difficulty of generating real-world 3D models with high-quality texture as well as robust pose estimation of occluded and symmetric objects. Evaluations with three challenging benchmark datasets show that Pix2Pose significantly outperforms state-of-the-art methods while solving these aforementioned problems.

Our results reveal that many failure cases are related to unseen poses that are not sufficiently covered by training images or the augmentation process. Therefore, future work will investigate strategies to improve data augmentation to more broadly cover pose variations using real images in order to improve estimation performance. Another avenue for future work is to generalize the approach to use a single network to estimate poses of various objects in a class that have similar geometry but different local shape or scale.

Acknowledgment The research leading to these results has received funding from the Austrian Science Foundation (FWF) under grant agreement No. I3967-N30 (BURG) and No. I3969-N30 (InDex), and Aelous Robotics, Inc.

References

[1] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *The European Conference on Computer Vision (ECCV)*, 2014. 2, 5

[2] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 6, 7

[3] Thanh-Toan Do, Trung Pham, Ming Cai, and Ian Reid. Real-time monocular object instance 6d pose estimation. In *British Machine Vision Conference (BMVC)*, 2018. 1, 2, 6

[4] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. 5

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2, 4

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 5

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[8] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision (ACCV)*, 2012. 2, 5, 6

[9] Tomáš Hodaň, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017. 2, 5, 7

[10] Tomáš Hodaň, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiri Matas, and Carsten Rother. Bop: Benchmark for 6d object pose estimation. In *The European Conference on Computer Vision (ECCV)*, 2018. 1, 6, 7

[11] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):107, 2017. 2

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 3

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2, 4

[14] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 6, 8

[15] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *The European Conference on Computer Vision (ECCV)*, 2016. 2, 7

[16] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015. 5

[17] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155, Jul 2008. 2, 5

[18] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *The European Conference on Computer Vision (ECCV)*, 2018. 1

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 5, 8

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *The European Conference on Computer Vision (ECCV)*, 2014. 5, 6

[21] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in rgb. In *The European Conference on Computer Vision (ECCV)*, 2018. 1

[22] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *The European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 6, 7

[23] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[24] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 6

[25] Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Feature mapping for learning fast and accurate 3d pose inference from synthetic images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 5, 8

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[28] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *The European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 3, 6, 7

[29] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 6

[30] Joel Vidal, Chyi-Yeu Lin, and Robert Martí. 6d pose estimation using an improved method based on point pair features. In *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, pages 405–409, 2018. 1

[31] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010. 2

[32] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems (RSS)*, 2018. 2, 6, 7

[33] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012. 2

[34] Sergey Zakharov, Benjamin Planche, Ziyan Wu, Andreas Hutter, Harald Kosch, and Slobodan Ilic. Keep it unreal: Bridging the realism gap for 2.5 d recognition with geometry priors only. In *International Conference on 3D Vision (3DV)*, 2018. 2

[35] Berk alli, Aaron Walsman, Arjun Singh, Siddhartha S. Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics and Automation Magazine*, 22:36–52, 2015. 1