sensors

Article

SynPo-Net—Accurate and Fast CNN-Based 6DoF Object Pose Estimation Using Synthetic Training

Yongzhi Su [1,*], Jason Rambach [2,*], Alain Pagani [2] and Didier Stricker [1,2]

1 TU Kaiserslautern, 67663 Kaiserslautern, Germany; Didier.Stricker@dfki.de
2 German Research Center for Artificial Intelligence (DFKI), 67663 Kaiserslautern, Germany; alain.pagani@dfki.de
* Correspondence: yongzhi.su@dfki.de (Y.S.); Jason.Rambach@dfki.de (J.R.)

Abstract: Estimation and tracking of 6DoF poses of objects in images is a challenging problem of great importance for robotic interaction and augmented reality. Recent approaches applying deep neural networks for pose estimation have shown encouraging results. However, most of them rely on training with real images of objects with severe limitations concerning ground truth pose acquisition, full coverage of possible poses, and training dataset scaling and generalization capability. This paper presents a novel approach using a Convolutional Neural Network (CNN) trained exclusively on single-channel Synthetic images of objects to regress 6DoF object Poses directly (**SynPo-Net**). The proposed SynPo-Net is a network architecture specifically designed for pose regression and a proposed domain adaptation scheme transforming real and synthetic images into an intermediate domain that is better fit for establishing correspondences. The extensive evaluation shows that our approach significantly outperforms the state-of-the-art using synthetic training in terms of both accuracy and speed. Our system can be used to estimate the 6DoF pose from a single frame, or be integrated into a tracking system to provide the initial pose.

Keywords: object pose estimation; convolutional neural networks; training with synthetic images; deep learning; domain adaptation; 6DoF object pose; 6DoF object tracking

check for updates

Citation: Su, Y.; Rambach, J.; Pagani, A.; Stricker, D. SynPo-Net—Accurate and Fast CNN-Based 6DoF Object Pose Estimation Using Synthetic Training. *Sensors* **2021**, *21*, 300. https://doi.org/10.3390/s21010300

Received: 11 August 2020
Accepted: 31 December 2020
Published: 5 January 2021

1. Introduction

Robotic interaction plays an essential role in automatic production, showing a significant increase in demand in recent years [1]. At the same time, Augmented Reality (AR) has shown great potential in tasks such as maintenance and training [2,3], proving its ability to improve the efficiency of cognitive tasks. 6 Degree-of-Freedom (6DoF) pose estimation and tracking is a crucial technology for AR and robotic grasping tasks and has therefore recently received increasing attention by the computer vision and robotics communities.

Approaches relying on depth images exclusively or in conjunction with RGB images have achieved admirable results over the last years [4,5]. Depth information enables a more reliable pose estimation for low-textured objects, especially under challenging lighting conditions. However, depth information, which can be obtained from stereo cameras or other sensors such as Time-of-Flight (ToF) cameras, is still a privilege of a small group of devices with specific cost and performance limitations.

In contrast, monocular camera setups are low-cost and more compact. They are already available on most current mobile devices. Therefore, pose estimation algorithms relying only on RGB image data are of great importance while posing significant challenges as well. Classical approaches with RGB images [2,6] extract hand-crafted features from images and use them in a predefined matching procedure. However, the gradient required for feature extraction is sensitive to motion blur. Moreover, typical features used in image processing, such as ORB features [7], have limitations in scaling, rotation and illumination variations of the targets. They also require target objects with strong edge features.

Deep learning based approaches and especially Convolutional Neural Networks (CNNs) have shown excellent results on many computer vision tasks, such as object detection and classification [8–10], image segmentation [11] or optical flow [12]. The works of Kendall et al. [13,14] were the first attempts to use CNNs for regression of 6DoF poses for place recognition and direct relocalization. After that, several learning based approaches followed, achieving good results on the object pose estimation problem [15–18]. The use of 3D pose refinement methods such as Iterative Closest Point (ICP) [19] or 2D methods [20,21] to improve the initial estimate appears to be of crucial importance for the performance of these pose estimation methods.

The training dataset is a critical factor for the performance of deep learning based methods since a large amount of representative data is required. For tasks such as image classification or object detection, ground truth can be easily manually labeled. Obtaining ground truth data of object 6DoF poses is a more challenging task. It requires dedicated setups, such as a robotic arm or a tracker with additional markers [4,22]. Such approaches are time-consuming and can only cover a limited variation of the object poses, scene illumination and background. Apart from that, the use of real data can negatively impact the ability of trained networks to generalize well in new environments.

Due to these reasons, the use of synthetic images rendered using 3D models of objects is very promising. Training with synthetic images simplifies creating datasets with a large number of images while obtaining ground truth of the 6DoF object pose is given directly by the rendering system. However, new challenges arise from the use of synthetic data since trained models need to overcome the representation gap between real and synthetic data. Thus, a further step of domain adaptation is necessary. Therefore, the domain adaptation problem in deep learning is a highly active field of research.

In our previous work [18], we introduced a novel approach to overcome the representation gap between synthetic and real images. We suggested using the pencil filter as an image processing step. Both synthetic and real images are transferred to the pencil filter domain before the image processing. In the work presented here we improve several parts of our pipeline, from network architecture to rotation representation and synthetic training data preparation to achieve a significant increase in accuracy that surpasses the current state-of-the-art as shown in an extensive experimental evaluation on different datasets. In detail, we propose the following novel contributions extending our previous work:

- A CNN network architecture specifically designed for increasing accuracy in pose estimation regression through the replacement of pooling layers with convolutional layers.
- The use of lie algebra rather than quaternions for angle representation and regression.
- An ablation study that quantitatively shows the positive effect of all main points of our proposed approach.
- An overall approach that outperforms the state-of-the-art in 6DoF object pose estimation under similar conditions (i.e., no depth images in training, training exclusively on synthetic images) while being computationally very efficient due to the revised network architecture.

Some examples of the estimated object pose can be seen in Figure 1. SynPo-Net can efficiently initialize a frame-to-frame tracking system like VisionLib [23] by providing an initial pose or relocalizing the system when tracking is lost.

The rest of this paper is organized as follows—in Section 2, we summarize existing work in object pose estimation and domain adaptation. In Section 3, we formulate the addressed problem of our work in detail. We introduce our approach in Section 4, discussing network architecture and training, dataset generation and domain adaptation. Subsequently, we present an extensive experimental evaluation of our approach and a comparison to the state-of-the-art in Section 5. Finally, we give concluding remarks in Section 6.

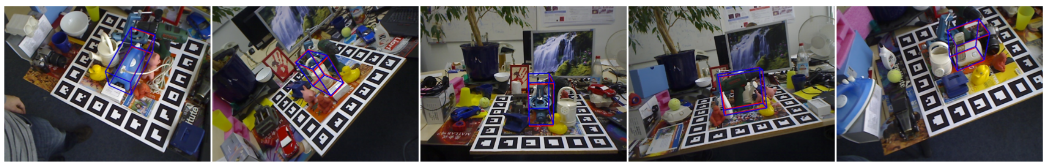

Figure 1. We visualize examples of the estimated pose using only SynPo-Net (without pose refinement). The groundtruth 3D bounding box and the predicted 3D bounding box are represented in red and blue, respectively.

2. Related Work

In this section, previous work related to our approach is classified and summarized. We first give a short overview of object pose estimation methods using depth and color information (RGB-D). Subsequently, we discuss state-of-the-art methods relying only on RGB images, which is directly comparable to our work. Additionally, we look at existing synthetic to real domain adaptation techniques not limited to pose estimation problems but also problems of learning from images in general.

2.1. RGB-D Object Pose Estimation

In classical approaches, 2D and 3D features are extracted from the RGB-D source, and hypotheses are made and verified to match the object 3D model in the scene. In template-based approaches, for example, in the work of Hinterstoisser et al. [4], templates are generated in different viewpoints of the object model. The template consists of color gradient features in the object contour and the depth gradient features on the object surface. The combination of both intensity and depth information helps to provide a reliable matching result. ICP [19] and its variations are often applied to refine the estimated pose. The accuracy and speed of these template-based methods are heavily dependant on the number of used templates. In Reference [24], sub-linear matching complexity was achieved. However, this usually trades speed for accuracy. Tejani et al. [25] adapted Reference [4] into a scale-invariant framework to reduce the number of templates. Point pair features based approaches [5,26] match local features instead of the whole template of the object. In this way, local details which may be discriminating will not be ignored. However, such methods appear to be computationally more demanding.

With the development of deep learning, the features for template matching pipeline can be learned with CNNs. In Reference [27], a CNN was used to extract the descriptors of the object from various viewpoints. The approach of Reference [28] achieved significant improvement by using learned local RGB-D features rather than the gradient. Furthermore, Reference [29] proposed a CNN and a multi-view fusion framework to leverage the information from multiple images, which has advantages, especially in video datasets. Additionally, the use of CNNs enables per-pixel matching or per-pixel prediction. In Reference [30], the proposed framework fuses both pixel-wise features from the image and point-wise features from the corresponding depth image. Predictions are made with each of those fused dense features, and the highest confidence pose is chosen as the final prediction. The work of Reference [31] can be seen as an extension of Reference [32] in the RGB-D case. It addressed the pose estimation problem by keypoint voting in the depth map. The pose can be calculated by fitting the detected 2D keypoints to their corresponding 3D keypoints in the object model.

2.2. RGB Object Pose Estimation

Most object pose estimation approaches using only RGB images are realized through deep neural networks. The approaches can be broadly classified into three categories.

The first category is the approach of extending object detection algorithms. Based on the detected 2D bounding box, various methods can be used to estimate the object rotation. In SSD-6D [15], the rotation is treated as a discrete viewpoint classification problem. Other methods directly regress rotation in the form of quaternions [33] or a lie algebra [34] representation. To deal with the occlusion problem, Sundermeyer et al. [17] proposed an autoencoder-decoder structure to determine the rotation relying on the object representation in the neural network. However, as Su et al. in Reference [35] point out, the appearance of the object depends not only on the rotation but also on the translation. Estimating the object rotation without considering the bounding box position is not accurate. Reference [36] later solved this issue by introducing a perspective correction.

Another group of approaches regress the object pose from the entire RGB image directly. A first attempt to use a CNN for regression of 6DoF poses was PoseNet [13]. The GoogLeNet [37] architecture was used for camera relocalization from images showing moderate accuracy but the method was not evaluated for object pose estimation. Following the idea of using a holistic CNN solution for pose estimation, in Reference [18], a similar network was applied for the regression of object poses. The pencil filter was used as a domain adaptation technique to enable training exclusively with synthetic images.

Finally, the third category of approaches determines 3D/2D point correspondences and solves a Perspective n Point (PnP) problem. In contrast to appearance-feature based keypoints, CNNs can detect keypoints in a more complex feature space. For instance, in the work of References [16,38], the 2D projections of the 3D bounding box corners are detected. However, the corners of 3D bounding boxes are virtual keypoints that physically do not belong to the object. In Reference [32], a CNN was trained to predict vectors pointing to the keypoints pixel-wise. A robust RANSAC based voting scheme was used to locate the 2D keypoints using these vectors. More recently, dense per pixel 2D-3D correspondences could be obtained. Park et al. [39] used an autoencoder-decoder to generate object masks with color to obtain the dense 2D-3D correspondences, with the RGB value representing the predicted position in the model local coordinate.

2.3. Domain Adaptation Techniques

Several methods have been introduced to deal with the domain adaptation problem when learning from synthetic images. Different methods are often combined in practice to obtain improved results.

Creating synthetic images that resemble reality as much as possible (photo-realistic rendering) is probably the most obvious solution to the problem [40,41]. However, the material of objects and lighting in complex conditions are not easy to simulate. To increase the realism, rendering the object context-aware is also very popular [42]. For example, the object should be positioned on a table. This is required to detect all the plane and its rotation in the background image. Therefore, such approaches are computationally expensive while they tend to perform well only in controlled environments.

Domain adaptation with the help of real images from the target domain is also common. Some techniques perform some form of post-processing to the synthetic data to increase the similarity to real images. Learning approaches can be used with pairs of synthetic and real images [43], or Generative Adversarial Networks (GANs) [44] can be trained to generate realistic images from synthetic images [45,46]. Small amounts of real images can also be used to fine-tune the CNN [42].

The appearance of the RGB images can be affected a lot by the environment. So the domain adaptation for the RGB images is inherently not easy. In contrast, depth images provide only spatial information. The domain gap between real depth images and synthetic depth images is much smaller than for RGB images. The texture and illuminations have minimal effect on the depth images. Rad et al. [47] use the features from the depth images to predict the pose. This part can be trained using synthetic datasets, since the domain gap between depth images can be easily overcome. A mapping from the depth map features and color images features can be trained using real RGBD images obtained from depth sensors. Georgakis et al. [48] also learns the key points with depth images, and then learns to match the color image features to the key point features from the depth image.

Furthermore, it is common to perform random data augmentation as domain randomization, for example, random noise, random brightness and contrast, random backgrounds to object views and random textures on objects [15,49]. To further improve the diversity of data, Reference [50] also changed the shape of 3D models to get more training images. Trained with images from different domains, the CNN is forced to focus on the real critical part of the image, which is not randomized, that is, objects in our case.

Unlike other works that attempt to fit one domain into another, we use a different approach to solve the domain adaptation problem in this work. We transform both the real and synthetic images into a new domain where visual similarity is increased and adaptation is facilitated (details in Section 4.2). Our approach is a general approach, we do not require any images or prior knowledge from the target domain. Our domain adaptation method is used together with domain randomization for further improvement.

3. Problem Formulation

The 6DoF object pose can be described with a rotation and a translation from the object coordinate system O to the camera coordinate system C. The translation part can be expressed with a translation vector $\boldsymbol{O_c} \in \mathbb{R}^3$ representing the position of the object coordinate system origin in the camera coordinate system. The rotation can be formulated in many different ways. In this work, we use lie algebra $\boldsymbol{\phi_{co}} \in \mathbb{R}^3$ with the footnote $\boldsymbol{co}$ denotes the rotation from object coordinate to camera coordinate.

In the frame of this work, we focus on the object pose estimation relying only on the color image, that is, given a single image, the pose of the target object should be estimated. Training of the proposed approaches is done exclusively with synthetic data.

4. Method

We describe the entire proposed pipeline of our object pose estimation system in this section. We first present the architecture of our SynPo-Net, which is the CNN designed for the task at hand, together with the used loss function in Section 4.1. Subsequently, we discuss how the predicted pose can be further refined, and the relationship between pose refinement and object 6DoF tracking in Section 4.1.3. In the end, we discuss the synthetic training data generation and the pencil filter as the proposed domain adaptation technique in Section 4.2).

4.1. 6DoF Object Pose Estimation

4.1.1. SynPo-Net Architecture

Currently, most neural network structures are designed for image classification tasks. In this paper, we argue that when using such kinds of network structures for object pose estimation, certain modifications can improve performance. Specifically, we used the Inception network [37] as a base network and researched the impact of input resolution, the use of pooling layers, and the representation of rotation.

Input Resolution: Li et al. [51] first mentioned that it is not suitable to directly use the object classification CNN as the backbone for object detection. Object classification tasks only need to recognize the object class. For this purpose, a global overview of the image is of high interest, which can be achieved by applying a large downsampling factor. However, object detection methods still need to estimate the object position accurately. The spatial resolution has more meaning in this case. This argument also applies to object pose estimation tasks. In Reference [51], the downsampling factor is reduced to double the output spatial resolution of the last CNN layers. Rather than reducing the downsampling factor, we suggest increasing the resolution of the input image.

Pooling Layers: Max pooling layers are widely used in CNNs for object classification [37,52]. To recognize object classes regardless of the object position in the image, these CNNs are required to be less sensitive to the object position. The max pooling layer selects only the maximum value in the reception field of the kernel. In other words, if the input layer has been shifted within half of the kernel size, the output layer does not change. This might be beneficial for classification tasks but not for pose regression that needs to be sensitive even to small changes of the target position.

To avoid the use of max pooling layers, we replace them with convolutional layers. More specifically, for max pooling layers followed by a convolutional layer, we merge them to one convolutional layer, in which the kernel size and the stride are the same as the max pooling layer and the number of output channels is the same as the initial convolutional layer (see Figure 2 as an example). For max pooling layers followed by inception blocks, we replace the max pooling layer with a convolutional layer without changing the kernel size, the stride, and the input size so that the output channel size of the convolutional layer equals the max pooling layer input channel size. We also replace the average pooling layers with convolutional layers. The convolutional layers can be equivalent to average pooling layers when the weight is learned as $1/(kernel\_size^2)$. So we suggest that this replacement can further increase the representative capacity of the network.

Representation of Rotation: Rotation matrices, Euler angles, quaternions and lie algebra are the most common representations of rotation. Rotation matrices can be used directly to rotate 3D points through matrix manipulation. However, using 9 elements to represent 3DoF transformations is unnecessarily redundant. Besides, rotation matrices need to be normalized, which introduces additional constraints to the optimization process.

Euler angles are easy to understand as a representation, and therefore commonly used for human-machine interaction. However, this representation is ambiguous, which means the same rotation can be represented with various combinations of Euler angles. Additionally, the gimbal lock problem creates essentially noncontinuous points during interpolation. These properties make Euler angles less suitable for optimization problems.

Quaternions are compact representations that consist of only 4 parameters and are unambiguous except that every quaternion is equal to the negative of itself. This representation also avoids the gimbal lock problem of Euler angles and allows a smooth interpolation for rotation [53]. Nevertheless, quaternions need to be normalized, which makes them suboptimal in regression tasks. (Details in Section 4.1.2).

Lie algebra $\mathbf{so}(\mathbf{3})$ is a representation of rotation extensively used in optimization problems. It is a compact 3-dimensional vector that can be mapped to a rotation matrix using the exponential map. At the same time, it is ambiguity free in an arbitrary $0 \sim 2\pi$ interval and does not require additional constraints. Therefore, we propose using lie algebra as a representation of rotation for regression with a CNN.

Other CNN Structure Adjustments: To make sure the number of output channels of the convolutional layers increases smoothly, we added more layers. Additionally, the technique of batch normalisation [54] has been applied to accelerate the training process, which was not used in our previous work.

Our proposed SynPo-Net is graphically represented in Figure 3.

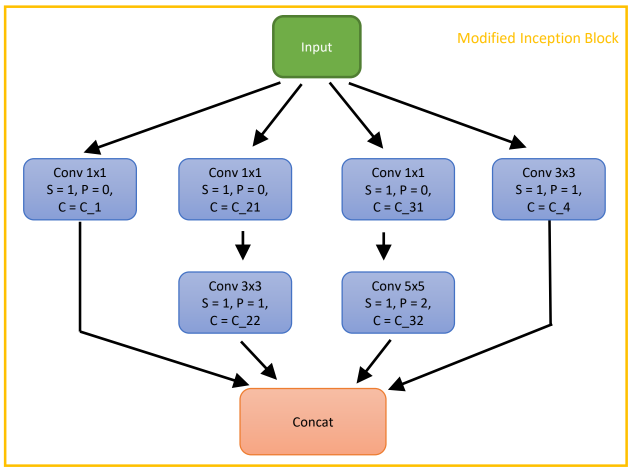
Modified Inception Block
Input
Conv 1x1 S = 1, P = 0, C = C_1
Conv 1x1 S = 1, P = 0, C = C_21
Conv 1x1 S = 1, P = 0, C = C_31
Conv 3x3 S = 1, P = 1, C = C_4
Conv 3x3 S = 1, P = 1, C = C_22
Conv 5x5 S = 1, P = 2, C = C_32
Concat

Figure 2. The proposed modified inception block. Each convolutional layer is followed by batch normalisation and a ReLU activation layer.

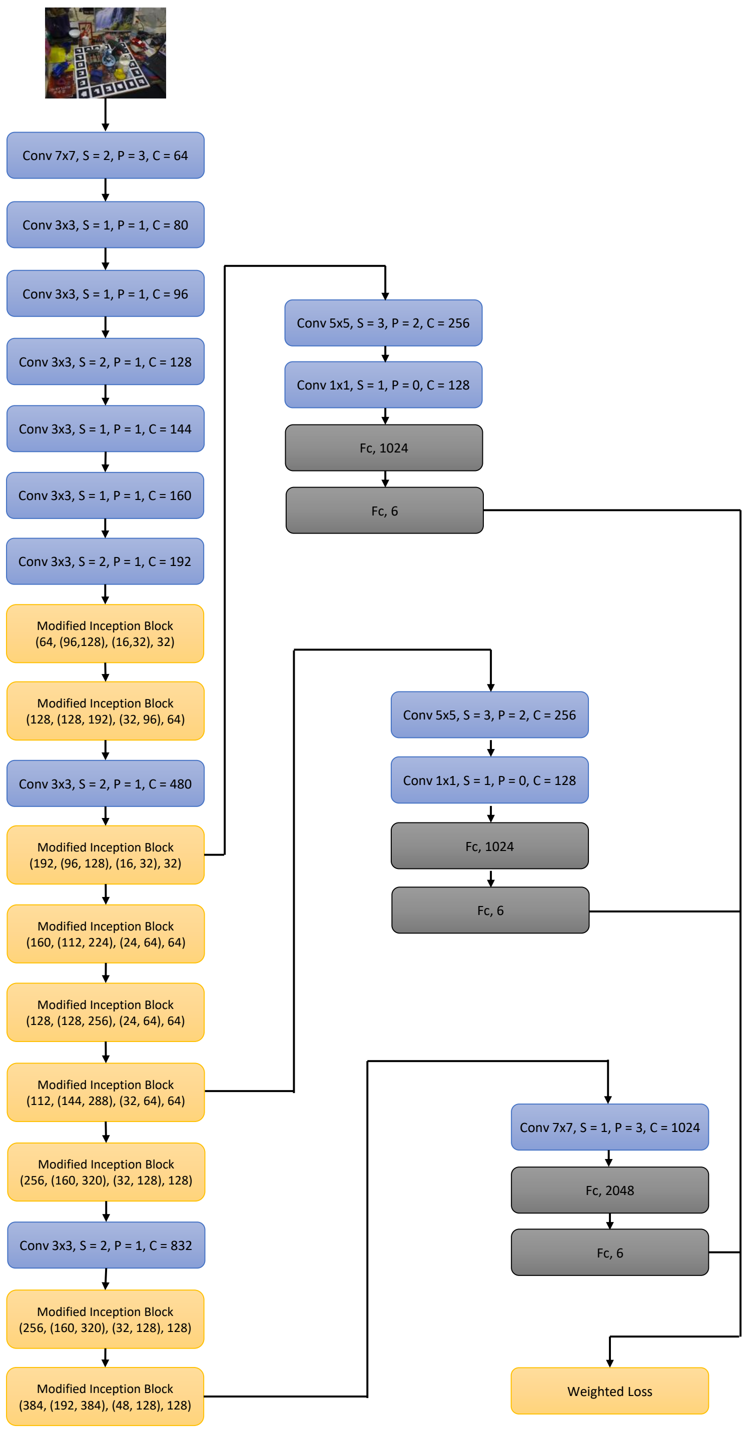
Conv 7x7, S = 2, P = 3, C = 64
Conv 3x3, S = 1, P = 1, C = 80
Conv 3x3, S = 1, P = 1, C = 96
Conv 3x3, S = 2, P = 1, C = 128
Conv 3x3, S = 1, P = 1, C = 144
Conv 3x3, S = 1, P = 1, C = 160
Conv 3x3, S = 2, P = 1, C = 192
Modified Inception Block (64, (96,128), (16,32), 32)
Modified Inception Block (128, (128, 192), (32, 96), 64)
Conv 3x3, S = 2, P = 1, C = 480
Modified Inception Block (192, (96, 128), (16, 32), 32)
Modified Inception Block (160, (112, 224), (24, 64), 64)
Modified Inception Block (128, (128, 256), (24, 64), 64)
Modified Inception Block (112, (144, 288), (32, 64), 64)
Modified Inception Block (256, (160, 320), (32, 128), 128)
Conv 3x3, S = 2, P = 1, C = 832
Modified Inception Block (256, (160, 320), (32, 128), 128)
Modified Inception Block (384, (192, 384), (48, 128), 128)
Conv 5x5, S = 3, P = 2, C = 256
Conv 1x1, S = 1, P = 0, C = 128
Fc, 1024
Fc, 6
Conv 5x5, S = 3, P = 2, C = 256
Conv 1x1, S = 1, P = 0, C = 128
Fc, 1024
Fc, 6
Conv 7x7, S = 1, P = 3, C = 1024
Fc, 2048
Fc, 6
Weighted Loss

Figure 3. The proposed SynPo-Net architecture. Each convolutional layer is followed by batch normalization and a ReLU activation layer. The 6 (or 7) pose values regressed by the Convolutional Neural Network (CNN) represent the 3D translation and 3D rotation vector of lie algebra (or quaternion). The architecture variant with quaternions is only used for the ablation study.

4.1.2. Loss Function Definition

We used L2-norm losses for both translation and rotation regression. In our previous work, we used quaternions to represent the rotation. In that case the loss function can be expressed as

$$L_{balanced\_O_c\_q} = \|O_c - \hat{O}_c\|_2 + \alpha_q \cdot \|q_{co} - \hat{q_{co}}\|_2 \,, \tag{1}$$

where $\boldsymbol{O_c}$ and $\boldsymbol{q_{co}}$ are the predicted translation vector and rotation quaternion and $\boldsymbol{\hat{O}_c}$ and $\boldsymbol{\hat{q_{co}}}$ are the respective ground truth values. Since the predicted quaternions are not restricted, we need to normalize them before they can be used to represent the rotation. α_q is the hyper-parameter used to balance the translation and rotation loss.

Using lie algebra to represent the rotation, the additional normalization can be avoided. Then the loss function can be formulated as

$$\boldsymbol{L_{balanced\_O_c\_lie}} = \|\boldsymbol{O_c} - \boldsymbol{\hat{O}_c}\|_2 + \alpha_l \cdot \|\boldsymbol{\phi_{co}} - \boldsymbol{\hat{\phi_{co}}}\|_2\,, \tag{2}$$

with $\boldsymbol{\phi_{co}}$ represent the predicted lie algebra rotation and $\boldsymbol{\hat{\phi_{co}}}$ the ground truth rotation. Thus, the loss function with lie algebra is more straightforward for optimizing the object rotation (without the normalization step). We used the $\boldsymbol{L_{balanced\_O_c\_lie}}$ to train SynPo-Net. Meanwhile, we also trained a CNN using the $\boldsymbol{L_{balanced\_O_c\_q}}$ only for comparison. The result can be found in the experimental section (see Section 5).

The loss functions discussed above is also calculated in the middle layers of the network as auxiliary losses and weighted into the primary loss of the network. Those auxiliary losses enable an effective gradient propagation in the lower layers and facilitate the training of the deep neural network. The weighted loss, which is used for the back propagation training, is defined as

$$\boldsymbol{L_{weighted}} = \gamma_1 \cdot \boldsymbol{L_{aux\_1}} + \gamma_2 \cdot \boldsymbol{L_{aux\_2}} + \gamma_3 \cdot \boldsymbol{L_{pri}}, \tag{3}$$

γ_1, γ_2, γ_3 are the hyper-parameters to adjust the effect of the auxiliary and primary loss.

4.1.3. Pose Refinement

Pose refinement is often used to improve the pose after an initial estimate is available, which can be optionally applied after the pose prediction from CNN. This task has a lot of similarities to frame-to-frame tracking. The tracking [21] or refinement [19,20] algorithm takes the information from previous frame or an initial pose and performs an improvement step of the estimate, often using geometry-based approaches. If the frame rate is high enough, we can expect that the pose difference between 2 continuous frames is minimal. In this case, the pose refinement algorithm can also be used for object tracking.

2D images are less sensitive to object depth translation than the object translation within the image plane. Thus, if depth is available, a pose refinement with depth information can help to estimate the pose very accurately. Since the LINEMOD dataset [4] also provides depth images. Similarly to other works, we also report the result with 3D refinement methods after using our proposed 2D-based estimation. We used the ICP algorithm for the pose refinement. Only the visible surface of the object is taken into account in each ICP iteration. The visible surface is updated after each iteration.

4.2. Training Dataset with Proposed Domain Adaptation Technique

A training dataset should cover as many as possible viewpoints of the object. Following the dataset generation pipeline described in Reference [18], we generated images of the objects from different viewpoints with random backgrounds from the PASCAL VOC [55] and the IKEA [56] datasets.

We augment our rendered training data by randomly adding various effects, that is, Gaussian noise, random contrast and brightness adjustment, motion blur, speckle noise (see Figure 4). In the previous work, the augmentations have been applied during the dataset generation. In this work, the augmentations are dynamically applied to the images every time they are loaded for training the network in a random way. Thus, the augmentations for a certain image are not fixed and the capacity of the dataset is increased further.

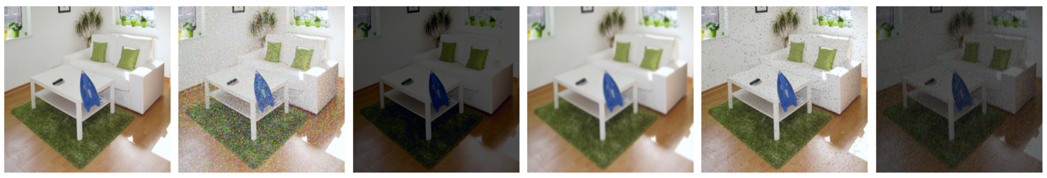

Figure 4. We apply randomly different kinds of effects on synthetic images before the application of the pencil filter. The examples of applied effects are shown from left to right: no effect, Gaussian noise, contrast and illumination changes, motion blur, speckle noise and mixture of all effects.

Subsequently, the pencil filter is applied on the synthetic training dataset and the resulting images are then used to train the network following the method of Reference [18]. Unlike other domain adaptation techniques, which attempt to transform one domain to another, we transform both domains into a third intermediate domain, in which the similarity between synthetic and real images is increased. We avoid providing color information to the network, which can be volatile when applied across datasets with different illumination conditions or between synthetic and real images. Also, unlike the 3D reconstructed models, the colors of CAD models are usually different from those of the final products. We use images in the pencil filter domain where the more reliable edge information is enhanced. In Figure 5, we present several rendered and real images with their corresponding pencil filter version to show the increased similarity in the pencil filter domain. This abstraction of information, apart from being effective in domain adaptation, also allows us to decrease the input size to our network from an RGB image to a single-channel image, positively influencing training and forward pass time.

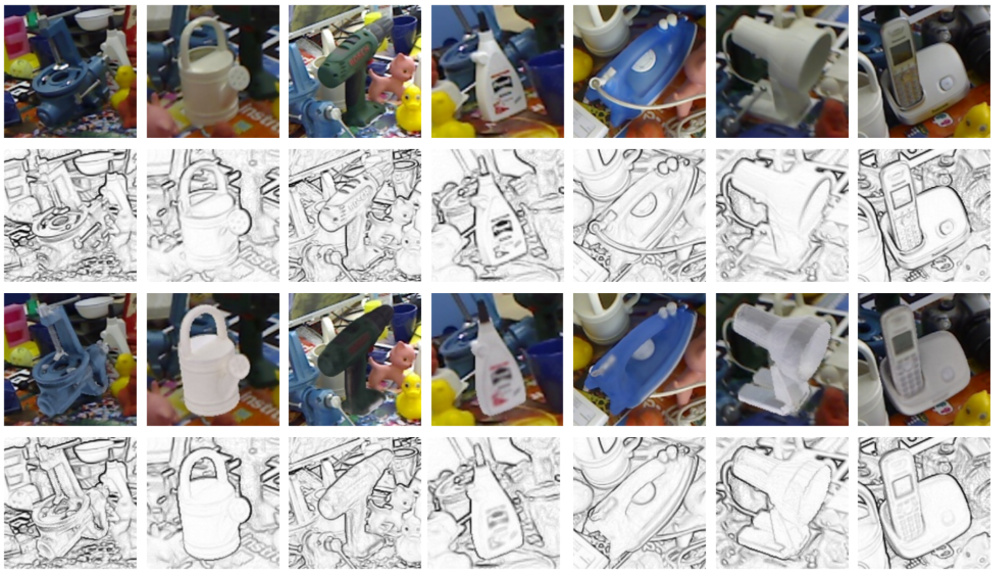

Figure 5. To visualize the effect of the pencil filter, we render the object model with the same pose over the real image. However, it should be noted that for training our network we only rendered the models on random backgrounds. First row: Cropped real images from the LINEMOD dataset [4]. Second row: Real images after applying pencil filter. Third row: Rendered images with same object pose and background as the first row. Fourth row: Rendered images after applying pencil filter.

5. Evaluation

Our evaluation results are presented in this section. We have performed an ablation study with selected objects from the LINEMOD dataset [4] to investigate the effects of each proposed CNN design and training decision separately. Subsequently, we compare against the state-of-the-art by evaluating our proposed CNN on the entire LINEMOD and TUD-L [57] datasets. LINEMOD is the most commonly used benchmark for object pose estimation and TUD-L is a dataset focusing specifically on lighting variations.

5.1. Implementation Details

We implemented our CNN with MXNet [58] and trained the CNN on an Nvidia GeForce GTX 2080Ti GPU (Nvidia, Santa Clara, CA, USA). We used the same CNN settings for evaluating on both datasets. We set the hyper-parameters α_q and α_l both equal to 30 to balance the loss of translation and rotation, and $\{\gamma_1, \gamma_2, \gamma_3\}$ as {10,10,40} to balance the effects of auxiliary losses and primary loss. We train our models using the ADAM optimizer [59] with a learning rate of 0.0002 and parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The models are trained for 800 epochs with a batch size of 16.

We created the training dataset with OpenGL. It consists of a number of 35,000–40,000 random poses per object. For the LINEMOD training dataset, we rendered the images without any light effect (the object's color is provided directly from the 3D Model). For the TUD-L training dataset, 30% of the images are rendered without any light effect. For the rest 70% images, we applied only diffuse reflection (according to Phong lighting model [60]) with a random light source position (no specular reflection). The pencil filter was applied for the domain adaptation to the training and evaluation datasets.

5.2. Error Metrics

The Average Distinguishable Distance (ADD) error has been first proposed in Reference [4]. This error calculates the average distance of model points projected to the camera domain using the predicted pose to the same model points projected using the ground truth pose. ADD errors are compared to a threshold that is based on the object size (largest diameter) for fairness.

For better dealing with ambiguous cases including symmetry and occlusions, the Visible Surface Discrepancy (VSD) error was proposed in Reference [61] and optimized in Reference [57]. It measures distance difference in the depth image using only the visible part of the object in the image. The Maximum Symmetry-Aware Surface Distance (MSSD) error introduced in Reference [62] indicates the chance of successful grasp with the robot arm by focusing on the maximum prediction error rather than the average error in ADD. In contrast, maximum symmetry-aware projection distance (MSPD) is more suitable to evaluate the RGB-only methods that are used in the Benchmark for 6DoF Object Pose Estimation (BOP) [63]. The model points are projected into the image plane for the measurement, which overcomes the RGB-only method's weakness. To take full advantage of the different metrics, a method's average performance with the VSD, MSSD, and MSPD is used as the bop performance score. In our experiments, we use suitable metrics that allow comparison to the related state-of-the-art works. More specifically, we use ADD error for the evaluation of LINEMOD dataset and use bop performance score for the evaluation of TUD-L dataset.

5.3. Ablation Study

5.3.1. The Pencil Filter Effect

In our previous work [18], we have already presented a qualitative proof that the accuracy can be improved by applying the pencil filter for domain adaptation. In this work, we try to intuitively represent the effect of pencil filter in reducing the difference between the real and synthetic images. We made the following experiment similar to in Reference [64].

We trained two networks separately with the same synthetic images of an object. One of them was trained with single-channel pencil images, and the other was trained with RGB images. Then we rendered the object above the real images with the corresponding ground truth pose. We test how the CNN is activated differently based on both types of images (the rendered images and the original real images). We passed both types of images to the network to observe the absolute differences in the feature map after the third modified inception block, based on which the first auxiliary pose is predicted.

We first calculate the average absolute difference of this layer with all the images for the camera (cam) and watering can (can) in the LINEMOD dataset. The difference can be qualitatively represented with Figure 6. It is obvious that the maximum difference of the synthetic image and the real image is smaller in the pencil domain.

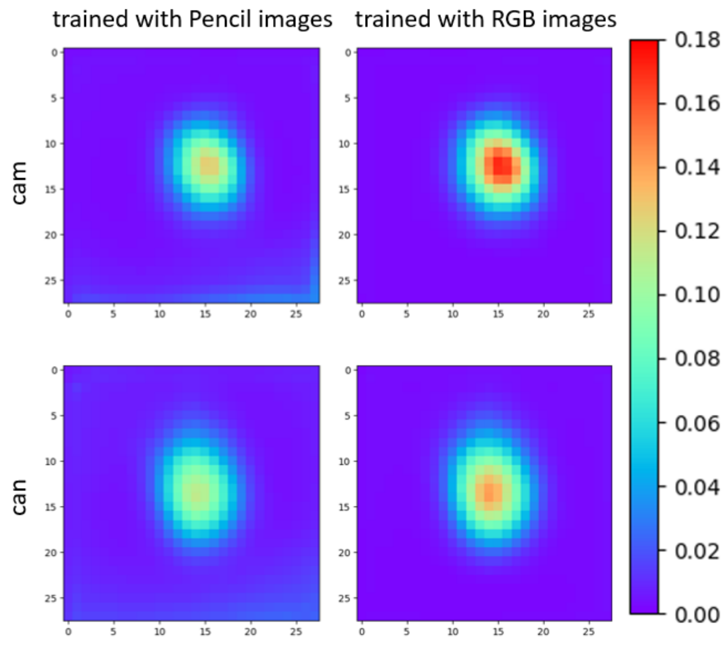

Figure 6. We compare the difference in activation of a CNN layer for a network trained with pencil images and with RGB images to illustrate the domain adaptation efficiency.

We also quantitatively report the {max absolute difference, mean absolute difference, standard deviation of absolute difference} in this averaged feature map of Figure 6. For the cam object trained with the pencil image, these values are {0.1254, 0.0126, 0.1146}, and trained with RGB images {0.1707, 0.0127, 0.1473} respectively. Despite the fact that pose estimation for the cam object has relatively low accuracy for our approach (see Section 5.4), the pencil filter still helps overcome the gap between the synthetic images and the real images. We achieved an outstanding result with the can object, and naturally, the absolute difference is even smaller than the case of the camera. For the network trained for the watering can, trained with pencil images, the values are {0.11309, 0.0159, 0.114}, and trained with RGB images {0.1414, 0.0137, 0.1398} respectively.

5.3.2. CNN Architecture Modification Effects

Our main ablation study results are presented in Table 1. Here, we evaluate the influence on the pose estimation accuracy for each one of the proposed ideas. We use the driller object of the LINEMOD dataset as an example and evaluate all proposed CNN modifications in this paper. The pencil filter was applied in all experiments. In the first five experiments, batch normalization was not used, and we set their learning rate to 0.0001 and batch-size of 32 to make full use of the GPU Memory. The CNN in the second experiment corresponds to our previous work [18]. The result in the second experiment is slight better than we reported in Reference [18], because we adjusted the {γ_1, γ_2, γ_3} values. The last experiment with all the modifications corresponds to our proposed SynPo-Net, the training settings are described in Section 5.1.

According to the Table 1, we can see that all the proposed modifications in this paper truly help in improving the CNN's performance for object pose estimation. Also, results indicate that the amount of data plays a crucial role. By applying random augmentations after the images have been loaded, we manage to increase the dataset diversity further and have a positive effect on the results.

Table 1. We evaluated the proposed contributions in an ablation study. The networks have been tested with the Driller object of the LINEMOD dataset using the Average Distinguishable Distance (ADD) metric with a threshold of 10%. Dynamic augmentation means that the random augmentations will be applied after the images have been loaded for training (as we also mentioned in the end of Section 4.2). The details of other modifications in the table are described in Section 4.1.1.

Input resolution (448 vs. 224)		√	√	√	√	√	√
Replace Pooling layers			√		√	√	√
Lie algebra				√	√	√	√
Dynamic Augmentation						√	√
Other CNN structure adjustments							√
ADD 10	14.06	15.57 (+1.51)	18.01 (+2.44)	19.95 (+1.94)	22.22 (+2.27)	41.75 (+19.53)	53.7 (+11.95)

5.4. LINEMOD Dataset State of the Art Comparison

We summarise the result of different methods on LINEMOD in Table 2 for comparison. The methods are divided into two groups based on the data type used for training. Our proposed CNN outperforms the state-of-the-art using synthetic training AAE [36] by a large margin for most objects and on average. However, their approach deals better with symmetric objects such as the glue and eggbox. Our approach removes color information and focuses more on edge information. The 3D model of the eggbox is coarse, which could also influence the quality of synthetic training datasets. Our CNN also performs weakly with the camera object. According to Figure 6, we can attribute the cause to the difference between the real image and synthetic image, as domain adaptation is not that successful in this case. Overall, our proposed method clearly shows the best-reported results thus far when training with synthetic data, and even exceeds Brachmann [65] and BB8 [16] which have been trained with real images. For further comparison, we also trained Pix2Pose [39] and YOLO6D [38] using the same synthetic images as ours (with all augmentations applied). For Pix2Pose [39], we provided the ground truth 2D detection bounding box. It is interesting to note that in contrast to the result when trained using real images, Pix2Pose [39] has a poor performance when trained using synthetic images. We think the dense-correspondence matching methods focus on the appearance of the object in pixel-level. Thus they are easier to overfit to the synthetic images and have problems generalizing to the real images, when trained solely using synthetic images.

Table 2. Evaluation results on the LINEMOD dataset using the ADD metric with a threshold of 10%, using RGB images only and no pose refinement. Higher is better. *: We trained YOLO6D [38] and Pix2Pose [39] using the same synthetic images as ours.

Training Data	**Synthetic Images**						**Real Images**				
Method	**SSD6D [15]**	**Rambach [18]**	**Pix2Pose * [39]**	**YOLO6D * [38]**	**AAE [36]**	**OURS**	**Brachmann [65]**	**BB8 [16]**	**YOLO6D [38]**	**Posecnn [33]**	**Pix2Pose [39]**
Ape	0.00	4.37	3.64	16.09	4.18	**23.14**	–	27.90	21.62	–	**58.10**
Benchvise	0.18	21.74	3.95	33.91	22.85	**75.23**	–	62.00	81.80	–	**91.10**
Cam	0.41	1.25	0.00	2.91	**32.91**	6.66	–	40.10	36.57	–	**60.90**
Can	1.35	2.09	16.55	20.98	37.03	**65.05**	–	48.10	68.80	–	**84.40**
Cat	0.51	2.54	20.18	27.14	18.68	**36.22**	–	45.20	41.82	–	**65.00**
Driller	2.58	12.46	28.96	24.66	24.81	**53.70**	–	58.60	63.51	–	**76.30**
Duck	0.00	4.78	0.23	20.17	5.86	**19.54**	–	32.80	27.23	–	**43.80**
Eggbox	8.90	1.43	0.00	2.31	**81.00**	3.86	–	40.00	69.58	–	**96.80**
Glue	0.00	7.38	7.29	15.00	**46.17**	41.80	–	27.00	**80.02**	–	79.40
Holepuncher	0.30	3.88	2.51	15.44	18.20	**21.10**	–	42.40	42.63	–	**74.80**
Iron	8.86	38.22	1.82	57.63	35.05	**85.07**	–	67.00	74.97	–	**83.40**
Lamp	8.20	27.35	30.15	26.75	61.15	**78.65**	–	39.90	71.11	–	**82.00**
Phone	0.18	5.39	31.94	14.80	36.27	**63.61**	–	35.20	47.74	–	**45.00**
Mean	2.42	10.22	11.32	21.43	32.63	**44.13**	32.30	43.60	55.95	62.7	**72.40**

In Table 3 we also report the results when ICP pose refinement is applied using depth images. Our result is better than that of Reference [36] after applying pose refinement as well. However, the projective ICP that was applied in Reference [15] takes leverage of both image and depth information and performs still best on average. This pose refinement approach is nevertheless not openly available for testing and experimentation. In any case, our approach still outperforms SSD-6D [15] on 9 out of 13 objects of the dataset.

Table 3. Results on LINEMOD dataset using the ADD metric with a threshold of 10%, when depth information is used for pose refinement. Higher is better.

Method		Ape	B.Vise	Cam	Can	Cat	Driller	Duck	E.Box	Glue	Holep.	Iron	Lamp	Phone	Mean
SSD6D [15]	+ P. ICP	65.00	80.00	78.00	86.00	70.00	73.00	**66.00**	**100.00**	**100.00**	49.00	78.00	73.00	79.00	**79.00**
AAE [36]	+ ICP	24.35	89.13	**82.10**	70.82	72.18	44.87	54.63	96.62	94.18	51.25	77.86	86.31	86.24	71.58
OURS	+ ICP	**65.86**	**94.98**	35.30	**94.82**	**79.81**	**83.50**	57.26	3.86	73.28	**68.63**	**96.18**	**94.70**	**91.59**	72.29

5.5. *TUD-L Dataset State of the Art Comparison*

The TUD-L dataset contains three household objects under challenging light conditions. We summarize the bop performance score [63] (as we described in Section 5.2) in Table 4. Object 2 (Frog) does not have an outstanding contour difference in different poses, and thus its pose is harder to be estimated than the other two objects. Still, our method shows superior performance against the other methods.

Table 4. Results on the TUD-L dataset using the bop performance score. Higher is better. (The result of AAE and Pixel2Pose are taken from bop website [63] on 31 July 2019).

Method	AAE [36]	Pixel2Pose [39]	OURS
Obj1 (Dragon)	–	–	47.35
Obj2 (Frog)	–	–	35.88
Obj3 (Watering Can)	–	–	59.80
mean	40.1	34.9	**47.67**

5.6. *Runtime Evaluation*

The frame processing rate achieved by state of the art methods is summarised in Table 5. The methods are tested with different hardware (GTX 1080 or Titan X Pascal). We used an RTX 2080 Ti, which is considered about 30% faster than Titan X Pascal [66]. Taken the hardware difference into consideration, our CNN should be able to run about $65/(1 + 30\%) = 50$ fps in Titan X. To summarize, we show that our CNN for pose estimation performs favorably against the state-of-the-art not only in terms of accuracy but also in terms of speed.

Table 5. Inference time of methods without refinement, according to Reference [36].

Method	fps
SSD6D [15]	12
AAE [36]	13 (RetinaNet) 42 (SSD)
BB8 [16]	4
Brachmann [65]	2
YOLO6D [38]	50
OURS	**65**

6. Conclusions

In this work, we proposed SynPo-Net, a novel CNN-based approach for 6DoF object pose estimation trained exclusively with RGB synthetic images reduced to single-channel images in pre-processing. We support the idea that neural network architectures need to be adjusted to the specific task of pose regression instead of relying on network layouts designed for classification. We address the domain adaptation problem by transforming synthetic and real images into a new domain with increased similarity. The results of an extensive experimental evaluation support our claims. In an ablation study, we showed how each proposed change to the network increases the pose accuracy. The comparison on the LINEMOD and TUD-L datasets proves that our method outperforms the existing state of the art in both accuracy and inference time. In future work, we plan to use our CNN as the backbone network for multi-object pose estimation.

Author Contributions: Conceptualization, Y.S. and J.R.; Data curation, Y.S. and J.R.; Formal analysis, Y.S. and J.R.; Funding acquisition, J.R. and D.S.; Investigation, Y.S.; Methodology, Y.S. and J.R.; Project administration, J.R. and A.P.; Resources, J.R.; Software, Y.S. and J.R.; Supervision, J.R., A.P. and D.S.; Validation, Y.S.; Writing—original draft, Y.S.; Writing—review & editing, J.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the INNOPROM Rheinland Pfalz/EFFRE funding program (P1-SZ2-7, 84002637) in cooperation with John Deere GmbH & Co. KG.

Acknowledgments: We are thankful to Xiaoying Tan for proofreading the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bahrin, M.A.K.; Othman, M.F.; Azli, N.N.; Talib, M.F. Industry 4.0: A review on industrial automation and robotic. *J. Teknol.* **2016**, *78*, 137–143.
2. Rambach, J.; Pagani, A.; Stricker, D. Augmented Things: Enhancing AR Applications leveraging the Internet of Things and Universal 3D Object Tracking. In Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR) 2017, Nantes, France, 9–13 October 2017.
3. Zhu, Z.; Branzoi, V.; Wolverton, M.; Murray, G.; Vitovitch, N.; Yarnall, L.; Acharya, G.; Samarasekera, S.; Kumar, R. AR-mentor: Augmented reality based mentoring system. In Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Munich, Germany, 10–12 September 2014; pp. 17–22.
4. 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 Proceedings of the Asian conference on computer vision (ACCV), Daejeon, Korea, 5–9 November 2012; pp. 548–562.
5. Vidal, J.; Lin, C.Y.; Martí, R. 6D pose estimation using an improved method based on point pair features. In Proceedings of the International Conference on Control, Automation and Robotics (ICCAR), Auckland, New Zealand, 20–23 April 2018; pp. 405–409.
6. Hinterstoisser, S.; Cagniart, C.; Ilic, S.; Sturm, P.; Navab, N.; Fua, P.; Lepetit, V. Gradient response maps for real-time detection of textureless objects. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 876–888. [CrossRef] [PubMed]
7. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 2564–2571.
8. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
9. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Annual Conference on Neural Information Processing Systems 2015, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
10. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
11. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef]
12. Dosovitskiy, A.; Fischer, P.; Ilg, E.; Hausser, P.; Hazirbas, C.; Golkov, V.; Van Der Smagt, P.; Cremers, D.; Brox, T. Flownet: Learning optical flow with convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 2758–2766.
13. Kendall, A.; Grimes, M.; Cipolla, R. Posenet: A convolutional network for real-time 6-dof camera relocalization. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 2938–2946.
14. Kendall, A.; Cipolla, R. Geometric loss functions for camera pose regression with deep learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5974–5983.
15. Kehl, W.; Manhardt, F.; Tombari, F.; Ilic, S.; Navab, N. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 22–29.
16. 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 Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; Volume 1, p. 5.
17. 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, Munich, Germany, 8–14 September 2018; pp. 699–715.

18. Rambach, J.; Deng, C.; Pagani, A.; Stricker, D. Learning 6dof object poses from synthetic single channel images. In Proceedings of the IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct) 2018, Munich, Germany, 16–20 October 2018; pp. 164–169.
19. Besl, P.J.; McKay, N.D. Method for registration of 3-D shapes. Sensor Fusion IV: Control Paradigms and Data Structures. *Int. Soc. Opt. Photonics* **1992**, *1611*, 586–607.
20. Manhardt, F.; Kehl, W.; Navab, N.; Tombari, F. Deep model-based 6d pose refinement in rgb. In Proceedings of the European Conference on Computer Vision (ECCV) 2018, Munich, Germany, 8–14 September 2018; pp. 800–815.
21. Drummond, T.; Cipolla, R. Real-time visual tracking of complex structures. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 932–946. [CrossRef]
22. Marion, P.; Florence, P.; Manuelli, L.; Tedrake, R. Label Fusion: A Pipeline for Generating Ground Truth Labels for Real RGBD Data of Cluttered Scenes. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2018, Brisbane, Australia, 21–25 May 2018; pp. 1–8.
23. Available online: https://visionlib.com/ (accessed on 1 March 2020).
24. Kehl, W.; Tombari, F.; Navab, N.; Ilic, S.; Lepetit, V. Hashmod: A hashing method for scalable 3D object detection. *arXiv* **2016**, arXiv:1607.06062.
25. Tejani, A.; Tang, D.; Kouskouridas, R.; Kim, T.K. Latent-class hough forests for 3D object detection and pose estimation. In Proceedings of the European Conference on Computer Vision (ECCV), Zurich, Switzerland, 6–12 September 2014; pp. 462–477.
26. Drost, B.; Ulrich, M.; Navab, N.; Ilic, S. Model globally, match locally: Efficient and robust 3D object recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, 13–18 June 2010; pp. 998–1005.
27. Wohlhart, P.; Lepetit, V. Learning descriptors for object recognition and 3d pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015, Boston, MA, USA, 7–12 June 2015; pp. 3109–3118.
28. Kehl, W.; Milletari, F.; Tombari, F.; Ilic, S.; Navab, N. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016; pp. 205–220.
29. Li, C.; Bai, J.; Hager, G.D. A unified framework for multi-view multi-class object pose estimation. In Proceedings of the European Conference on Computer Vision (ECCV) 2018, Munich, Germany, 8–14 September 2018; pp. 254–269.
30. Wang, C.; Xu, D.; Zhu, Y.; Martín-Martín, R.; Lu, C.; Li, F.; Savarese, S. Densefusion: 6d object pose estimation by iterative dense fusion. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2019, Long Beach, CA, USA, 16–20 June 2019; pp. 3343–3352.
31. He, Y.; Sun, W.; Huang, H.; Liu, J.; Fan, H.; Sun, J. PVN3D: A Deep Point-wise 3D Keypoints Voting Network for 6DoF Pose Estimation. *arXiv* **2019**, arXiv:1911.04231.
32. Peng, S.; Liu, Y.; Huang, Q.; Zhou, X.; Bao, H. Pvnet: Pixel-wise voting network for 6dof pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2019, Long Beach, CA, USA, 16–20 June 2019; pp. 4561–4570.
33. Xiang, Y.; Schmidt, T.; Narayanan, V.; Fox, D. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *arXiv* **2017**, arXiv:1711.00199.
34. Do, T.T.; Pham, T.; Cai, M.; Reid, I. Real-time monocular object instance 6d pose estimation. In Proceedings of the British Machine Vision Conference (BMVC), Newcastle, UK, 3–6 September 2018; Volume 1, p. 6.
35. Su, Y.; Rambach, J.; Minaskan, N.; Lesur, P.; Pagani, A.; Stricker, D. Deep Multi-state Object Pose Estimation for Augmented Reality Assembly. In Proceedings of the IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Beijing, China, 10–18 October 2019; pp. 222–227.
36. Sundermeyer, M.; Marton, Z.C.; Durner, M.; Triebel, R. Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection. *Int. J. Comput. Vis.* **2020**, *128*, 714–729. [CrossRef]
37. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
38. Tekin, B.; Sinha, S.N.; Fua, P. Real-time seamless single shot 6d object pose prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018, Salt Lake City, UT, USA, 18–22 June 2018; pp. 292–301.
39. Park, K.; Patten, T.; Vincze, M. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In Proceedings of the IEEE International Conference on Computer Vision 2019, Seoul, Korea, 27–28 October 2019; pp. 7668–7677.
40. Mitash, C.; Bekris, K.; Boularias, A. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 545–551.
41. Movshovitz-Attias, Y.; Kanade, T.; Sheikh, Y. How useful is photo-realistic rendering for visual learning? In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016; pp. 202–217.
42. Wang, H.; Sridhar, S.; Huang, J.; Valentin, J.; Song, S.; Guibas, L.J. Normalized object coordinate space for category-level 6d object pose and size estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2019, Long Beach, CA, USA, 16–20 June 2019; pp. 2642–2651.
43. Csurka, G. Domain adaptation for visual applications: A comprehensive survey. *arXiv* **2017**, arXiv:1702.05374.

44. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Annual Conference on Neural Information Processing Systems 2014, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
45. Bousmalis, K.; Silberman, N.; Dohan, D.; Erhan, D.; Krishnan, D. Unsupervised pixel-level domain adaptation with generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 3722–3731.
46. Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; Webb, R. Learning from simulated and unsupervised images through adversarial training. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2107–2116.
47. Rad, M.; Oberweger, M.; Lepetit, V. Domain transfer for 3d pose estimation from color images without manual annotations. In Proceedings of the Asian Conference on Computer Vision 2018, Perth, Australia, 2–6 December 2018; pp. 69–84.
48. Georgakis, G.; Karanam, S.; Wu, Z.; Kosecka, J. Learning local rgb-to-cad correspondences for object pose estimation. In Proceedings of the IEEE International Conference on Computer Vision 2019, Seoul, Korea, 27–28 October 2019; pp. 8967–8976.
49. DeTone, D.; Malisiewicz, T.; Rabinovich, A. Toward geometric deep SLAM. *arXiv* **2017**, arXiv:1707.07410.
50. Su, H.; Qi, C.R.; Li, Y.; Guibas, L.J. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 2686–2694.
51. Li, Z.; Peng, C.; Yu, G.; Zhang, X.; Deng, Y.; Sun, J. Detnet: A backbone network for object detection. *arXiv* **2018**, arXiv:1804.06215.
52. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
53. Dam, E.B.; Koch, M.; Lillholm, M. *Quaternions, Interpolation and Animation*; Datalogisk Institut, Københavns Universitet: Copenhagen, Denmark, 1998; Volume 2.
54. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv* **2015**, arXiv:1502.03167.
55. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
56. Lim, J.J.; Pirsiavash, H.; Torralba, A. Parsing ikea objects: Fine pose estimation. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, 1–8 December 2013; pp. 2992–2999.
57. Hodan, T.; Michel, F.; Brachmann, E.; Kehl, W.; GlentBuch, A.; Kraft, D.; Drost, B.; Vidal, J.; Ihrke, S.; Zabulis, X.; et al. BOP: Benchmark for 6D object pose estimation. In Proceedings of the European Conference on Computer Vision (ECCV) 2018, Munich, Germany, 8–14 September 2018; pp. 19–34.
58. Available online: https://mxnet.apache.org/ (accessed on 1 March 2020).
59. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
60. Phong, B.T. Illumination for computer generated pictures. *Commun. ACM* **1975**, *18*, 311–317. [CrossRef]
61. Hodaň, T.; Matas, J.; Obdržálek, Š. On evaluation of 6D object pose estimation. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016; pp. 606–619.
62. Drost, B.; Ulrich, M.; Bergmann, P.; Hartinger, P.; Steger, C. Introducing mvtec itodd-a dataset for 3d object recognition in industry. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Venice, Italy, 22–29 October 2017; pp. 2200–2208.
63. Available online: https://bop.felk.cvut.cz/home/ (accessed on 1 March 2020).
64. Rad, M.; Oberweger, M.; Lepetit, V. Feature Mapping for Learning Fast and Accurate 3D Pose Inference from Synthetic Images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4663–4672.
65. Brachmann, E.; Michel, F.; Krull, A.; Ying Yang, M.; Gumhold, S. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 3364–3372.
66. Available online: https://gpu.userbenchmark.com/Compare/Nvidia-Titan-X-Pascal-vs-Nvidia-RTX-2080-Ti/m158352vs4027 (accessed on 1 March 2020).