

---


# ITERATIVE OPTIMISATION WITH AN INNOVATION CNN FOR POSE REFINEMENT

---

PREPRINT\*

 **Gerard Kennedy**

Systems Theory and Robotics Group  
Australian National University  
ACT, 2601, Australia  
Gerard.Kennedy@anu.edu.au

 **Zheyu Zhuang**

Systems Theory and Robotics Group  
Australian National University  
ACT, 2601, Australia  
Zheyu.Zhuang@anu.edu.au

 **Xin Yu**

University of Technology  
NSW, 2007, Australia  
Xin.Yu@uts.edu.au

 **Robert Mahony**

Systems Theory and Robotics Group  
Australian National University  
ACT, 2601, Australia  
Robert.Mahony@anu.edu.au

January 22, 2021

## ABSTRACT

Object pose estimation from a single RGB image is a challenging problem due to variable lighting conditions and viewpoint changes. The most accurate pose estimation networks implement pose refinement via reprojection of a known, textured 3D model, however, such methods cannot be applied without high quality 3D models of the observed objects. In this work we propose an approach, namely an Innovation CNN, to object pose estimation refinement that overcomes the requirement for reprojecting a textured 3D model. Our approach improves initial pose estimation progressively by applying the Innovation CNN iteratively in a stochastic gradient descent (SGD) framework. We evaluate our method on the popular LINEMOD and Occlusion LINEMOD datasets and obtain state-of-the-art performance on both datasets.

**Keywords** pose estimation, robotic vision, deep learning

## 1 Introduction

The task of object pose estimation involves estimating the 6D pose (translation and orientation) of a specified object relative to a specified reference frame. This is an important problem in computer vision, and has a range of applications including robotic manipulation/grasping, and virtual/augmented reality. Recent state-of-the-art RGB pose estimation algorithms use a two phase approach, initially providing a rough estimate of pose, and then using a second network to refine the pose estimate in order to obtain the desired performance [KMT<sup>+</sup>17, ZSI19]. The principle of refining an existing estimate of pose can be applied multiple times [LWJ<sup>+</sup>19a, ZSI19, MKNT18]. Such an approach can be seen as analogous to iterative state refinement algorithms such as stochastic gradient descent (SGD) that are well established in the optimisation field [Fle87].

---

\*Under review

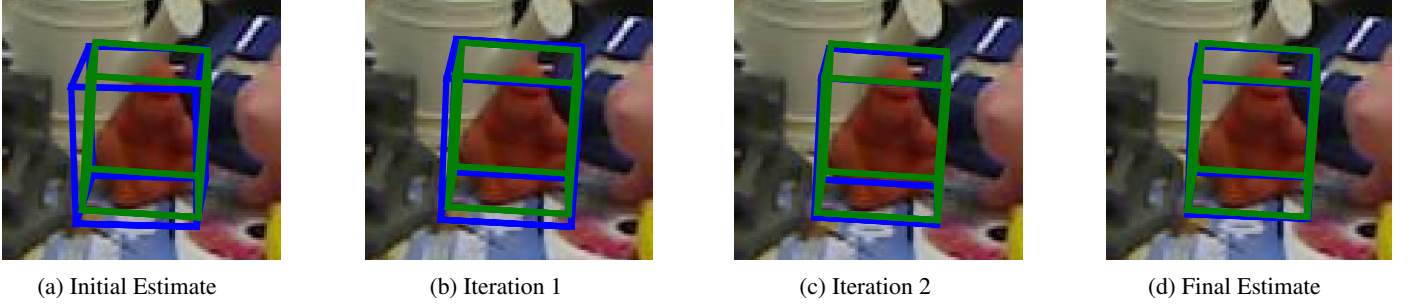


Figure 1: Our proposed method applied to the Ape object of the **LINEMOD** dataset. **Green** bounding boxes represent ground truth poses and **blue** boxes represent our results.

In this paper, we draw from the established principles of SGD and formulate a novel CNN architecture to implement an iterative refinement based on formal principles. In doing so, we ease the requirement for the neural network to accurately estimate the pose in a single forward pass. Specifically, we modify the loss function for a classical pose estimation network to estimate an update or refinement to an existing state, rather than estimating the state directly. An autoencoder network is implemented in parallel with the modified pose estimation network architecture to encode the state estimate and inject this information into the modified pose estimation network using skip connections. This overcomes the requirement to reconstruct an image from the state information in order for the state information to be injected. The output of the modified pose estimation network is used as an SGD update for the state estimate. Motivated by filtering theory we label this output the *innovation*. In the filtering field the innovation refers to the difference between the predicted and measured information. We therefore call our algorithm the *Innovation CNN*. By formally recognising the role of the state estimate and the SGD update term, we are able to employ all the existing algorithms and insights on step-size choice and convergence analysis that have been developed for SGD.

We evaluate our approach on two widely used benchmarks for 6D object pose estimation, LINEMOD [HKN] and Occlusion LINEMOD [BKM<sup>+</sup>14]. Our approach refines initial pose estimates, leading to an average improvement of 3.69% on LINEMOD and 3.31% on Occlusion LINEMOD in terms of the widely-used ADD(-S) metric. This leads us to achieve state-of-the-art results on both datasets. We further demonstrate that our method is particularly effective at refining relatively poor initial estimates. Specifically, we show relatively large improvement on the more difficult Occlusion LINEMOD dataset, along with particularly challenging objects of the LINEMOD dataset, such as ape (21.14% improvement) and duck (10.73% improvement) where the objects have poor surface texture.

In summary, the key contributions of this paper are:

- A novel framework that uses a CNN to estimate a state update term and utilise this to refine an initial estimate via SGD.
- A demonstration that this framework improves the results of an off-the-shelf object pose estimation network.
- State-of-the-art performance on the **LINEMOD** and **Occlusion LINEMOD** datasets<sup>1</sup>.

## 2 Related Work

Object pose estimation literature can be sub-divided into four general areas: end-to-end regression from image to pose, pose classification via a discretised pose space, regression to an intermediate representation such as keypoints, and pose refinement. In this section we briefly review literature related to object pose estimation from a single RGB image based on these four categories.

### 2.1 Pose Estimation

**End-to-End Regression:** End-to-end pose regression from a single image is a highly challenging task due to the nonlinearity of the space of rigid 3D rotations  $SO(3)$ . This is addressed in [KGC15] by separating and carefully balancing translation and rotation terms in the loss function, although this is for camera pose estimation, a slightly different problem. More recently, [XSNF17] decouple or ‘disentangle’ translation and rotation terms by moving the centre of rotation to the centre of the object and representing translation in 2D image space. In practice, regression to 3D orientation has had limited success [MAV17, SMD<sup>+</sup>19]. This is partly due to such methods only covering the small subsection of pose space that is seen during training [KMT<sup>+</sup>17], which has in turn led to discretisation of the pose space as discussed below. Recently, [WMS<sup>+</sup>20] obtained state-of-the-art results on real data by separately regressing rotation and translation within a self-supervised framework that also utilises RGBD images.

**Pose Classification:** Pose classification has typically involved discretising  $SO(3)$ , while the translation component is obtained via regression. However, as even a coarse discretisation of  $SO(3)$  ( $\sim 5^\circ$  precision) leads to over 50,000 possible cases [SMD<sup>+</sup>19].

<sup>1</sup>Recent work by Yu *et al.* [YZKL20] now outperforms our results.

Kehl *et al.* [KMT<sup>+</sup>17] approach this problem by decomposing 6D pose into viewpoint and discretised in-plane rotation. However, as noted in [SMD<sup>+</sup>19], this approach leads to ambiguous classifications as a change of viewpoint can be nearly equivalent to a change of in-plane rotation. Furthermore, such classification leads to coarse pose estimates that require further refinement [KMT<sup>+</sup>17, LWJ<sup>+</sup>19a], as discussed below.

**Pose via an Intermediate Representation:** A recent, popular approach to pose estimation is to first regress to an intermediate representation such as keypoints, from which pose can be obtained via 2D-3D correspondences and a PnP algorithm [TSF17, RL17, PLH<sup>+</sup>19, SSH20, ZSI19, IWJ19b, HBM20]. Of these approaches, some, including [TSF17, RL17] regress to a set of bounding box corners. Such methods encounter difficulty when objects are occluded or truncated. Alternatively, [PLH<sup>+</sup>19] predicts pixel-wise unit-vectors that in turn indicate direction to keypoints. Song *et al.* [SSH20] implements these vector directions from [PLH<sup>+</sup>19] while also estimating object edge vectors and symmetry correspondences. Zakharov *et al.* [ZSI19] apply texture to objects with a 2D image generated from spherical or cylindrical projections, and uses these to estimate dense correspondences. Li *et al.* [IWJ19b] apply this intermediate representation approach to estimate rotation, but estimates translation separately via regression.

## 2.2 Pose Refinement

Of the networks discussed above, several have additional refinement steps that can be added to improve results. For example, DeepIM [LWJ<sup>+</sup>19a] is presented as a refinement step for POSECNN [XSNF17], the work [MKNT18] is presented as a refinement step for SSD6D [KMT<sup>+</sup>17], and DPOD [ZSI19] is presented along with a refinement step. In each case a synthetic image of the target object is rendered using the initial pose estimate and a 3D model of the target object.

The key difference in these approaches is in the representation of the relative pose. DeepIM [LWJ<sup>+</sup>19a] utilise an ‘untangled representation’ in which the centre of the object is the centre of rotation, and translation is estimated in pixel space. CosyPose [LCAS20] improve upon DeepIM by utilising a more recent feature detection network, and implement their framework to estimate object poses in a multi-view scene. Manhardt *et al.* [MKNT18] represent rotation as a unit quaternion and translation as a vector in  $\mathbb{R}^3$ . DPOD [ZSI19] combines these approaches by estimating rotation relative to the object centre, and estimating translation as a vector in  $\mathbb{R}^3$ . These approaches therefore differ mainly in their loss functions, and all require textured 3D object models.

## 3 Proposed Innovation CNN

In this section we present a formulation for a general state estimation algorithm and apply this formulation to the problem of object pose estimation. We also present our network architecture, training loss, and network implementation.

### 3.1 State Estimation

Motivated by filtering and optimisation principles [AM79], we treat object pose estimation as an offline state estimation task. In this context a *state* is an internal representation of a system that is sufficient to fully define the future evolution of all system variables. A *state estimation* algorithm is a dynamical system that takes measurements from the true system as inputs and whose state is an estimate of the true system state.

Consider a dynamical system in a state  $\mathbf{X}$  defined by

$$\dot{\mathbf{X}}(t) = f(\mathbf{X}(t), \mathbf{V}(t)), \quad (1)$$

$$\mathbf{y}(t) = h(\mathbf{X}(t)), \quad (2)$$

where  $f(\cdot)$  is the state model  $\mathbf{V}(t)$  is an measured input signal,  $h : \mathbf{X}(t) \rightarrow \mathbf{y}(t)$  is an output map, and  $t$  represents the current time. We think of the output  $\mathbf{y}(t)$  as encoding the specific target information that the engineer is interested in, while the state  $\mathbf{X}$  encodes the full information necessary for the propagation of the dynamical system.

The class of estimation algorithm considered in this paper are of the form

$$\dot{\hat{\mathbf{X}}}(t) = f(\hat{\mathbf{X}}(t), \mathbf{V}(t)) - \Delta(t), \quad (3)$$

$$\hat{\mathbf{y}}(t) = h(\hat{\mathbf{X}}(t)), \quad (4)$$

where  $\Delta$  is the innovation term that is chosen so as to drive the estimated outputs towards the true outputs, and  $\hat{\cdot}$  represents an estimation.

### 3.2 Object Pose Refinement via State Estimation

The proposed approach involves taking the output state from an existing object pose estimation network and refining this output using a state estimation framework to obtain more accurate pose estimates. We choose PVNet [PLH<sup>+</sup>19] to be the baseline

object pose estimation network. In established algorithms such as PVNet, the state is estimated directly via a deep neural network. A typical formulation for such a problem can be written

$$\widehat{\mathbf{X}} = \mathcal{E}(\mathbf{V}) \quad (5)$$

$$\widehat{\mathbf{y}} = h(\widehat{\mathbf{X}}), \quad (6)$$

where the estimator  $\mathcal{E}$  is implemented as a CNN or more classical computer vision algorithm.

The output of PVNet is a unit vector field, with vectors pointing from each pixel to each of a set of keypoints. The vectors are defined to be

$$\boldsymbol{\eta}_{ij}^k = \boldsymbol{\xi}^k - \boldsymbol{\xi}_{ij}, \quad (7)$$

where  $\boldsymbol{\xi}_{ij}$  denotes a 2D pixel with coordinates  $(i, j)$  within an image  $I$  with dimensions  $M \times N$ , and  $\boldsymbol{\xi}^k$  represents the pixel location of keypoint  $k \in \mathcal{K}$ . The unit vector field is

$$\mathbf{X}_{ij}^k = \frac{\boldsymbol{\eta}_{ij}^k}{\|\boldsymbol{\eta}_{ij}^k\|_2} \in \mathbb{R}^{2 \times \mathcal{K} \times M \times N}. \quad (8)$$

This unit vector field forms the state for the estimation problem.

We assume a static state model

$$f(\mathbf{X}(t), \mathbf{V}(t)) = 0. \quad (9)$$

Approximating the time derivative by  $\dot{\mathbf{X}} = \delta t(\mathbf{X}(t) - \mathbf{X}(t-1))$ , the corresponding state estimation algorithm has the form

$$\widehat{\mathbf{X}}(t) = \widehat{\mathbf{X}}(t-1) - \Delta(t-1), \quad (10)$$

$$\widehat{\mathbf{y}}(t) = h(\widehat{\mathbf{X}}(t)), \quad (11)$$

where  $\delta t$  is absorbed into the innovation  $\Delta$ .

The output of the state estimation algorithm  $\widehat{\mathbf{y}}$  is the object pose. This is a discrete system with time steps  $t \in \{0, 1, \dots, T\}$ .

The function  $h$  maps a vector field representation of keypoints to object pose via a RANSAC and uncertainty-driven PnP framework (EPnP), as discussed in [PLH<sup>+</sup>19]. The task then becomes to learn an appropriate  $\Delta$ , such that the error in the state estimation algorithm is iteratively driven towards zero.

Consider the standard L2-norm loss function

$$\Phi(t) = \frac{1}{2} \|\widehat{\mathbf{X}}_{ij}^k(t) - \mathbf{X}_{ij}^k\|_2^2. \quad (12)$$

The gradient of this function is

$$\begin{aligned} \nabla \Phi(t) &= \frac{1}{2} \nabla_{\widehat{\mathbf{X}}_{ij}^k(t)} \|\widehat{\mathbf{X}}_{ij}^k(t) - \mathbf{X}_{ij}^k\|_2^2 \\ &= \widehat{\mathbf{X}}_{ij}^k(t) - \mathbf{X}_{ij}^k, \end{aligned} \quad (13)$$

where  $\mathbf{X}_{ij}^k$  is the ground truth vector field, and  $\nabla_x$  denotes the gradient operator with respect to  $x$ . Set

$$\Delta(t) = \alpha(t) \nabla \Phi(t), \quad (14)$$

where  $\alpha(t)$  is a sequence of step-sizes that must be specified.

In practice, the gradient term  $\nabla \Phi(t)$  is not directly measured. Instead, we use an estimate of this value,

$$\widehat{\nabla \Phi}(t) = \mathcal{F}(\mathbf{V}, \widehat{\mathbf{X}}(t))$$

where  $\mathcal{F}$  represents the output of the CNN described in Section 3.3.

The iterative update is applied by employing gradient descent

$$\widehat{\mathbf{X}}_{ij}^k(t+1) = \widehat{\mathbf{X}}_{ij}^k(t) - \alpha(t) \widehat{\nabla \Phi}(t). \quad (15)$$

Within this iterative framework the initial state  $\widehat{\mathbf{X}}_{ij}^k(0)$  is the vector field estimate obtained from PVNet, and  $\widehat{\nabla \Phi}(t)$  is estimated by our network. Algorithm 1 summarises the proposed information pipeline.

**Algorithm 1** Iterative Optimisation with Innovation CNN

- 1: Choose  $0 < \alpha(t) < 1$
  - 2: Choose  $T > 0$
  - 3: Input:  $I, \mathbf{P}^k$
  - 4:  $\widehat{\mathbf{X}}_{ij}^k(0) \leftarrow \text{PVNet}(I)$
  - 5: **for**  $t = 1 \rightarrow T$  **do**
  - 6:    $\widehat{\nabla\Phi}(t) \leftarrow \text{Innov}(I, \widehat{\mathbf{X}}_{ij}^k(t))$
  - 7:    $\widehat{\mathbf{X}}_{ij}^k(t+1) = \widehat{\mathbf{X}}_{ij}^k(t) - \alpha(t)\widehat{\nabla\Phi}(t)$
  - 8:  $\widehat{\mathbf{y}} = \text{EPnP}(\widehat{\mathbf{X}}_{ij}^k(T), \mathbf{P}^k)$
  - 9: Output:  $\widehat{\mathbf{y}}$
- ▷ Maximum #iterations  
 ▷ Image, 3D object points  
 ▷ Pose

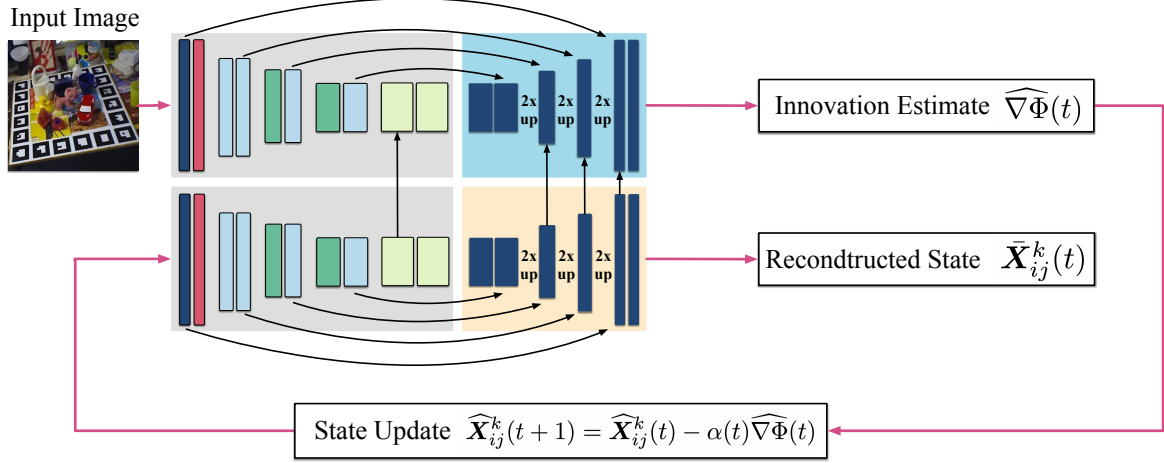


Figure 2: Network Architecture: Innovation Estimator (top block), and Estimate Autoencoder (bottom block). The input image (top left) is passed to the Innovation Estimator, which estimates  $\widehat{\nabla\Phi}(t)$  (top right). The current state estimate  $\widehat{\mathbf{X}}_{ij}^k(t)$  is passed to the State Encoder (bottom left). The output of the State Decoder (bottom right) is  $\bar{\mathbf{X}}_{ij}^k(t)$ , which represents the same information as  $\widehat{\mathbf{X}}_{ij}^k(t)$ . Using  $\widehat{\nabla\Phi}(t)$  the state  $\widehat{\mathbf{X}}_{ij}^k$  is updated via gradient descent (Eq. (15)). The updated state estimate  $\widehat{\mathbf{X}}_{ij}^k(t+1)$  is then passed to the State Decoder in an iterative process.

### 3.3 Network Architecture

The network consists of two encoder-decoder blocks operating in parallel (see Figure 2). The top block in Figure 2 is labeled the *Innovation Estimator*, and the bottom block is labeled the *Estimate Autoencoder*. The Innovation Encoder consists of a pre-trained ResNet-18 [HZRS16] architecture. The Innovation Decoder consists of successive convolution and upsampling operations. The Estimate Autoencoder consists of a pre-trained ResNet-18 backbone followed by three successive upsampling layers. The input dimensionality of the Estimate Autoencoder modified to accept a vector field state estimate.

The Innovation Estimator receives an image and returns an estimate of the gradient of the vector field state  $\widehat{\nabla\Phi}$ . The Estimate Autoencoder receives a vector field state  $\widehat{\mathbf{X}}_{ij}^k$  and returns an estimate of the vector field state  $\bar{\mathbf{X}}_{ij}^k$ . The Estimate Autoencoder is therefore an auxiliary task aimed at allowing information from the initial state estimate to be injected into the Innovation Estimator via skip connections. This auxiliary task leads to improvement on the main task by leveraging domain specific information via multi-task learning.

### 3.4 Training Loss

The state loss is applied to the Estimate Autoencoder, and is defined by

$$\mathcal{L}_{\text{state}} = \sum_{k=1}^{\mathcal{K}} \sum_{(i,j) \in \mathcal{S}} \|\widehat{\mathbf{X}}_{ij}^k(t) - \bar{\mathbf{X}}_{ij}^k(t)\|_1^2, \quad (16)$$

where  $(i, j) \in \mathcal{S}$  indicates that the pixel is within the segmentation mask, and the norm that is used is the ‘smooth l1 norm’ for unit vectors proposed by [Gir15]. The state  $\bar{\mathbf{X}}_{ij}^k$  represents the same information as  $\widehat{\mathbf{X}}_{ij}^k$  and is used only for the purpose of training the autoencoder.

The state gradient loss is applied to the Innovation Estimator, and is defined by

$$\mathcal{L}_{\text{innov}} = \sum_{k=1}^{\mathcal{K}} \sum_{(i,j) \in \mathcal{S}} \|\widehat{\nabla \Phi}(t) - \nabla \Phi(t)\|_1^2 \quad (17)$$

$$= \sum_{k=1}^{\mathcal{K}} \sum_{(i,j) \in \mathcal{S}} \|\widehat{\nabla \Phi}(t) - (\widehat{\mathbf{X}}_{ij}^k(t) - \mathbf{X}_{ij}^k)\|_1^2. \quad (18)$$

The loss is backpropogated after each iteration of the gradient descent defined by Eq. (15).

The loss function used to train the network is

$$\mathcal{L} = \mathcal{L}_{\text{innov}} + \gamma \mathcal{L}_{\text{state}}, \quad (19)$$

where  $\gamma$  is a scaling factor.

### 3.5 Implementation Details

The network is trained for  $T = 2$  iterations of Eq. (15) at each epoch, with a constant step size of  $\alpha = 0.6$ . We use a pretrained PVNet [PLH<sup>+</sup>19] to provide the initial estimate  $\mathbf{X}_{ij}^k(0)$  for each object. We use a batch size of 64, and down-sample training images by four to fit the learning algorithm on our machine. We compute 8 keypoints via farthest point sampling, from which object pose is obtained via uncertainty-driven PnP [PLH<sup>+</sup>19]. We also render 10,000 images and synthesis 10,000 images for each object. We employ data augmentation in the form of random cropping, resizing, rotation, colour jittering. An Adam optimizer [KB15] is employed with an initial learning rate of  $1e^{-3}$ , which is decayed to  $1e^{-5}$  by a factor of 0.85 every 10 epochs. The learning hyperparameter in Eq. (19) is set to  $\gamma = 10$ . We train our network for 50 epochs.

During testing, we employ Hough voting to localise keypoints, before using uncertainty-driven PnP to obtain object pose from keypoints. We set the step size to  $\alpha = 0.01$  and perform iterations of Eq. (13) until the estimate converges. Once the gradient output of our Innovation CNN approaches zero we consider the estimate to be converged. Figure 4 provides qualitative results that illustrate this iterative convergence.

## 4 Experiments

In this section we conduct experiments on two widely used datasets for object pose estimation, and evaluate our performance with two standard metrics for this task.

### 4.1 Datasets

We conduct experiments on two popular datasets for object pose estimation.

**LINEMOD** [HKN] consists of 15783 images of 13 objects, with approximately 1200 instances for each object. Challenging aspects of this dataset include lighting variations, scene clutter, object occlusions, and texture-less objects.

**Occlusion LINEMOD** [BKM<sup>+</sup>14] is a subset of LINEMOD images with each image containing multiple annotated objects under severe occlusion, thus presenting a more challenging scenario for accurate pose estimation.

### 4.2 Evaluation Metrics

We evaluate our approach using two standard metrics for object pose estimation.

The **ADD** metric [HKN] computes the average 3D distance between the points of the 3D model under transformation from the ground truth and estimated pose respectively. Specifically, given the ground truth rotation  $\mathbf{R}$  and translation  $\mathbf{T}$  and the estimated rotation  $\hat{\mathbf{R}}$  and translation  $\hat{\mathbf{T}}$ , the ADD metric computes

$$\text{ADD} = \frac{1}{m} \sum_{\mathbf{x} \in \mathcal{M}} \|(\mathbf{R}\mathbf{x} + \mathbf{T}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{T}})\|_2, \quad (20)$$

where  $\mathcal{M}$  denotes the set of 3D model points and  $m$  is the number of points. For symmetric objects we use the similar **ADD(-S)** metric [XSNF17], where the mean distance is computed based on the closest point distance,

$$\text{ADD(S)} = \frac{1}{m} \sum_{\mathbf{x}_1 \in \mathcal{M}} \min_{\mathbf{x}_2 \in \mathcal{M}} \|(\mathbf{R}\mathbf{x}_1 + \mathbf{T}) - (\hat{\mathbf{R}}\mathbf{x}_2 + \hat{\mathbf{T}})\|_2. \quad (21)$$

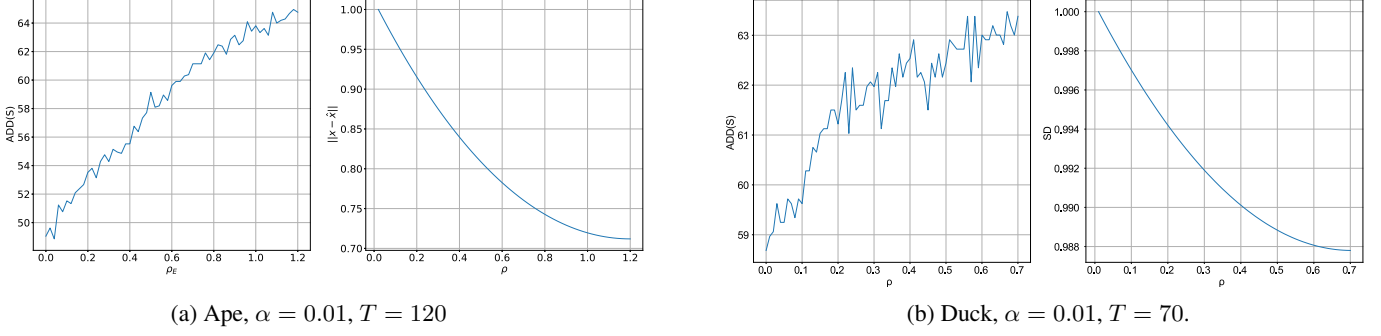


Figure 4: Iterative improvement on the **ADD(-S)** metric and corresponding percentage decrease in the State Distance (SD) for the ape and duck objects of the **LINEMOD** dataset plotted against the interpolation distance  $\rho$  (Eq. (24)).

We denote both metric as ADD(-S) and use the one appropriate to the object. In each case a pose is considered correct if the average distance is less than 10% of the 3D model diameter. The reported metric is the percentage of poses that are considered correct.

The **2D projection** metric [HKN] computes the mean distance between pixel locations of the 2D projections of the 3D model model, when projection is performed with the estimated and ground truth poses. The metric is defined via

$$\text{2d Proj} = \frac{1}{|V|} \sum_{\mathbf{v} \in V} \|\mathbf{P}\mathbf{Y}\mathbf{v} - \mathbf{P}\hat{\mathbf{Y}}\mathbf{v}\|_2, \quad (22)$$

where  $V$  is the set of all object model vertices,  $\mathbf{Y}$  is the pose, and  $\mathbf{P}$  is the camera matrix. The estimated pose is considered correct if the average error is less than 5 pixels. The metric reported is the percentage of correct poses.

Finally, we propose an additional metric, dubbed the *State Distance* (SD), designed to quantify the difference between the estimated and ground truth vector field states. This is used to provide an indication of whether the state estimate  $\hat{\mathbf{X}}_{ij}^k$  is converging to or diverging from the true state  $\mathbf{X}_{ij}^k$ . This metric is defined by

$$\text{SD} = \frac{1}{s} \sum_{(i,j) \in \mathcal{S}} \|\hat{\mathbf{X}}_{ij}^k(t) - \mathbf{X}_{ij}^k\|_2^2, \quad (23)$$

where  $(i, j) \in \mathcal{S}$  indicates that the pixel is within the segmentation mask, and  $s$  denotes the number of pixels within the segmentation mask.

### 4.3 Comparison to State-of-the-Art

Results in terms of the ADD(-S) and 2D projection metrics on the LINEMOD and Occlusion LINEMOD datasets are presented in Tables 1, 2, and 3. Qualitative results illustrating our final pose estimates are provided in Figures 3 and 5.

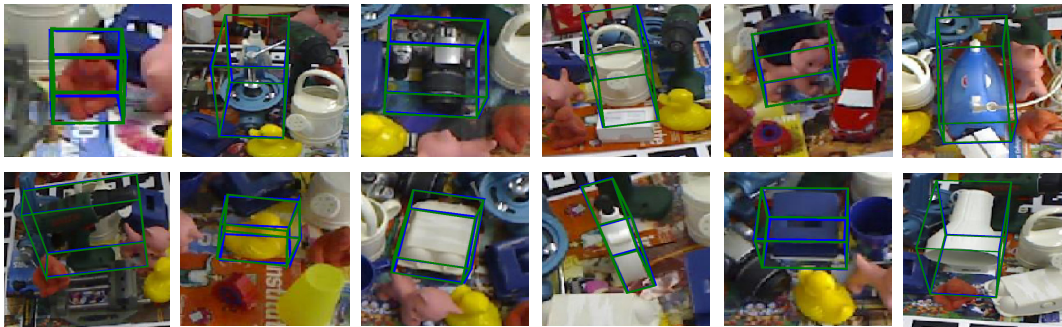


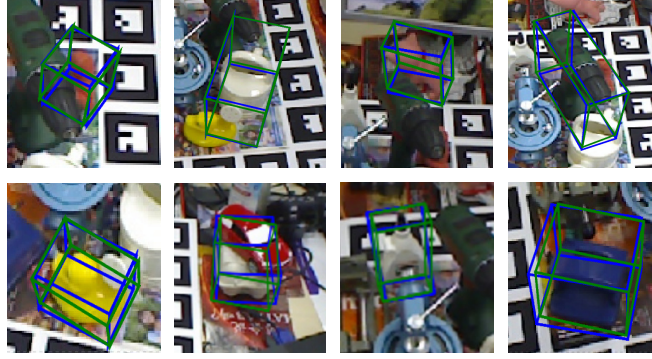
Figure 3: Visualisation of our qualitative results on the **LINEMOD** dataset. **Green** bounding boxes represent ground truth poses and **blue** boxes represent our results.



Methods	CDPN [IWJ19b]	YOLO6D [RDGF16]	PVNet [PLH <sup>+</sup> 19]	OURS
ape	92.10	96.86	<b>99.23</b>	<b>99.23</b>
benchwise	95.06	98.35	<b>99.81</b>	<b>99.81</b>
cam	93.24	98.73	<b>99.21</b>	99.12
can	97.44	99.41	<b>99.90</b>	99.70
cat	97.41	99.80	99.30	<b>99.70</b>
driller	79.41	95.34	96.92	<b>97.23</b>
duck	94.65	<b>98.59</b>	98.02	98.02
eggbox	90.33	98.97	<b>99.34</b>	99.24
glue	96.53	<b>99.23</b>	98.45	98.45
holepuncher	92.86	99.71	<b>100.0</b>	99.70
iron	82.94	97.24	<b>99.18</b>	<b>99.18</b>
lamp	79.87	95.49	<b>98.27</b>	98.18
phone	86.07	97.64	<b>99.42</b>	<b>99.42</b>
<b>average</b>	90.37	98.10	<b>99.00</b>	<b>99.00</b>

Table 2: Performance comparison on the **LINEMOD** dataset with respect to the **2D Projection** error.

Methods	DPOD [ZSI19]	Pix2Pose [PPV19]	PVNet [PLH <sup>+</sup> 19]	OURS
ape	-	22.0	15.81	<b>26.41</b>
can	-	44.7	<b>63.30</b>	61.31
cat	-	<b>22.7</b>	16.68	19.88
driller	-	44.7	65.65	<b>70.10</b>
duck	-	15.0	25.24	<b>31.99</b>
eggbox	-	25.2	<b>50.17</b>	49.44
glue	-	32.4	49.62	<b>51.16</b>
holepuncher	-	<b>49.5</b>	39.67	42.34
<b>average</b>	32.79	32.0	40.77	<b>44.08</b>

Table 3: Performance comparison on the **Occlusion LINEMOD** dataset with respect to the **ADD(-S)** metric.Figure 5: Visualisation of our qualitative results on each object of the **Occlusion LINEMOD** dataset. **Green** bounding boxes represent ground truth poses and **blue** boxes represent our results.

Methods	DPOD [ZSI19]	CDPN [IWJ19b]	PVNet [PLH <sup>+</sup> 19]	OURS
ape	53.28	64.38	43.62	<b>64.76</b>
benchwise	95.34	97.77	<b>99.90</b>	<b>99.90</b>
cam	90.36	91.67	86.86	<b>92.58</b>
can	94.10	95.87	95.47	<b>96.56</b>
cat	60.38	<b>83.83</b>	79.34	82.83
driller	<b>97.72</b>	96.23	96.43	97.52
duck	66.01	<b>66.76</b>	52.58	63.31
eggbox	<b>99.72</b>	99.72	99.15	99.32
glue	93.83	<b>99.61</b>	95.66	96.91
holepuncher	65.83	<b>85.82</b>	81.92	82.20
iron	<b>99.80</b>	97.85	98.88	99.31
lamp	88.11	97.86	99.33	<b>99.42</b>
phone	74.24	90.75	92.41	<b>94.33</b>
<b>average</b>	82.98	89.86	86.27	<b>89.92</b>

Table 1: Performance comparison on the **LINEMOD** dataset with respect to the **ADD(-S)** metric.



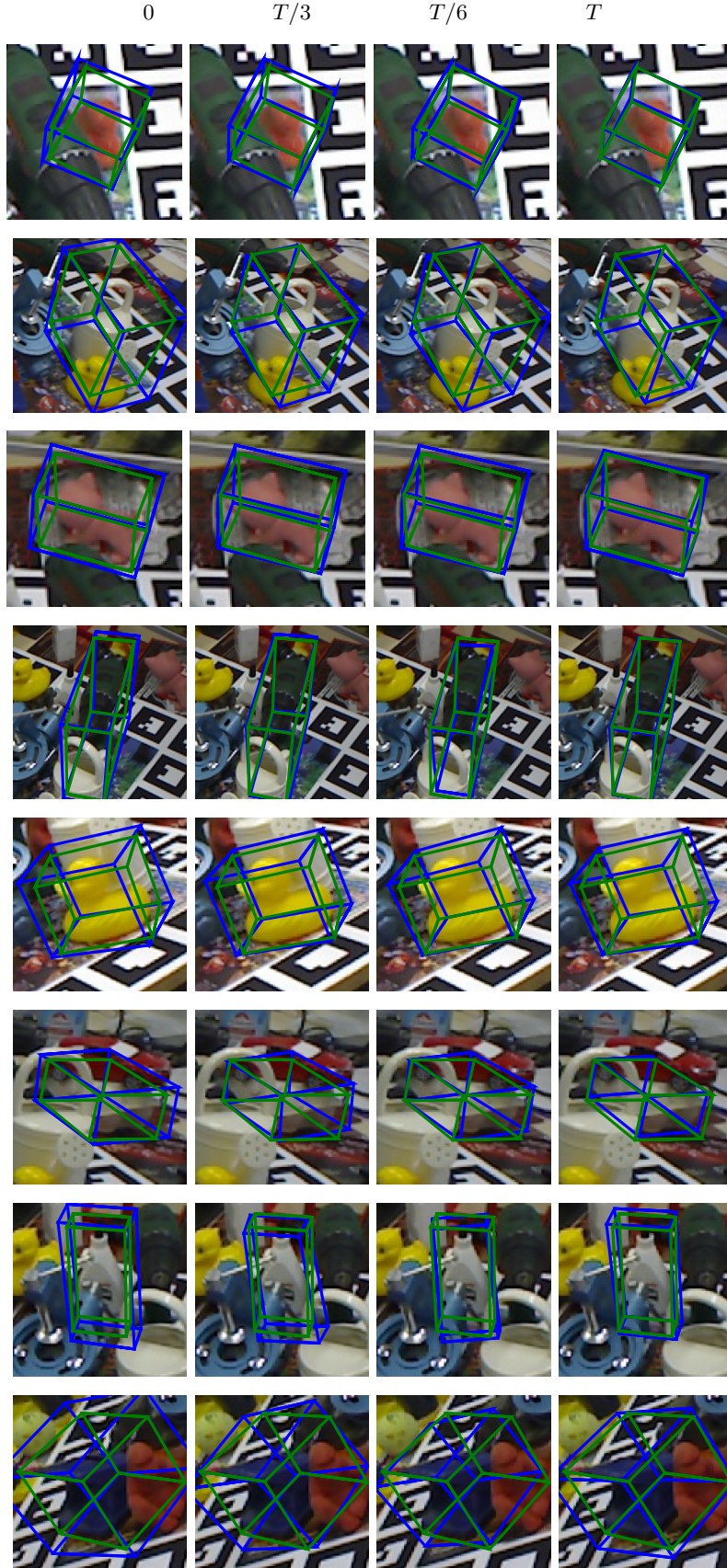


Figure 6: Iterative pose refinement on all objects in the **Occlusion LINEMOD** dataset. The initial pose estimate is shown in the left ( $t = 0$ ), and the final pose estimate is shown on the right ( $t = T$ ), with equidistant intermediate poses shown in-between. **Green** bounding boxes represent ground truth poses and **blue** boxes represent our results.

In this paper we focus on refining an initial pose estimate without leveraging any extra supervision. In Tables 1,2,3 we compare our results to the top three other pose estimation networks in this category. Our approach outperforms all previous methods and achieves state-of-the-art performance on both the LINEMOD and Occlusion LINEMOD datasets.

Most significantly, our network generates pose estimates that lead to a higher average on both the ADD(-S) and 2D projection metrics, compared to the baseline network. This includes a performance increase of 3.69% on the LINEMOD dataset and 3.31% on the Occlusion LINEMOD dataset in terms of the ADD(-S) metric. This is equivalent to improving the performance by 4.2% and 8.12% respectively compared to the performance of the baseline network.

It can be seen that the largest improvement is obtained for objects for which the baseline network provides a relatively poor initial estimate. Such objects include the relatively texture-less ‘ape’ and ‘duck’ and all objects in the Occlusion LINEMOD dataset. Specifically, on the ape and duck objects we improve baseline estimates by 21.14% and 10.73% respectively in terms of the ADD(-S) metric. Qualitative results for these two objects are presented in Figure 4, for which the ADD(-S) metric is evaluated at each iteration of Eq. (15). Figure 4 also illustrates the iterative convergence of the state distance SD. In this figure, the ADD(S) and SD metrics are plotted as a function of the *interpolation distance*,

$$\rho = \alpha(t)T. \quad (24)$$

The interpolation distance quantifies how far we have iterated from the initial estimate.

Qualitative results illustrating the iterative application of our method are presented for the Occlusion LINEMOD dataset in Figure 6. To generate Figure 6 the initial and final pose are displayed, along with two intermediate poses sampled from the iterative framework. For example, for the Ape object  $T = 120$  iterations were used, and Figure 6 displays poses for  $T=\{0,40,80,120\}$ .

#### 4.4 Ablation Study

We conducted experiments with a range of configurations before choosing the configuration used to produce the presented results. Results of these experiments are summarised in Table 4. In Table 4, column one shows a comparison of training with (✓) and without (✗) the Estimate Autoencoder. Column two shows a comparison of backpropagating each iteration of Eq. (15) (✓) and backpropagating after  $T$  iterations (✗). Column three shows a comparison of choosing the output of PVNet as  $\mathbf{X}_{ij}^k(0)$  (✓) and choosing the ground truth vector field computed from keypoints perturbed with 1% noise (✗). All ablation experiments were conducted on the Ape object of the LINEMOD dataset.

Results shown are the mean **ADD(-S)** results obtained from testing the final 20 epochs of the model. In Table 4 we compare the results of experiments with and without the Estimate Autoencoder (the bottom network block in Figure 2). We also compare our backpropagation scheme (backprop each iteration of Eq. (15)) with the alternative scheme of accumulating the loss and backpropagating after  $T$  iterations, as discussed in [HR19]. Finally we compare the impact of using a pretrained PVNet as the initial estimate  $\mathbf{X}_{ij}^k(0)$  against using the ground truth vector field, or the ground truth vector field perturbed with different levels of noise. We trialed a range of different noise levels, and the result of applying 1% noise is presented in Table 4.

	Estimate Autoencoder	Backprop each iteration	PVNet Initial Estimate
✓	<b>53.93</b>	52.01	<b>53.93</b>
✗	50.61	<b>53.93</b>	46.00

Table 4: Ablation Study of key design choices. The study indicates that the State Autoencoder improves performance, that it is better to backpropagate after each iteration of Eq. (15), and that it is better to use the output of PVNet as the initial estimate during evaluation.

## 5 Conclusion

In this paper we provide a new framework to improve an initial pose estimate. We present a novel CNN architecture capable of estimating the gradient from an initial state prediction to the true state. Our Innovation CNN refines initial state estimates in an SGD framework iteratively, thus greatly reducing the difficulty of estimating object poses in a single forward pass. Extensive experiments on widely used datasets demonstrate that we improve initial object pose estimates significantly and obtain state-of-the-art performance. Moreover, our Innovation CNN is generic but not restricted to our baseline method and we will apply it to different baseline networks in our future work.

## Acknowledgements

This research was supported in part by the Australian Government Research Training Program Scholarship and in part by the Australian Research Council through the “Australian Centre of Excellence for Robotic Vision” CE140100016.

## References

- [AM79] BDO Anderson and JB Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ, 1979.
- [BKM<sup>+</sup>14] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*. Springer, September 2014.
- [Fle87] R. Fletcher. *Practical Methods of Optimization; (2nd Ed.)*. Wiley-Interscience, USA, 1987.
- [Gir15] R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.
- [HBM20] Tomas Hodan, Daniel Barath, and Jiri Matas. Epos: Estimating 6d pose of objects with symmetries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [HKN] Stefan Hinterstoisser, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes.
- [HR19] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. *CoRR*, abs/1904.05290, 2019.
- [HZRS16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [KGC15] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946, 2015.
- [KMT<sup>+</sup>17] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1530–1538, 2017.
- [LCAS20] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [LWJ<sup>+</sup>19a] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. *International Journal of Computer Vision*, 128(3):657678, Nov 2019.
- [IWI19b] Zhigang li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *ICCV*, 10 2019.
- [MAV17] S. Mahendran, H. Ali, and R. Vidal. 3d pose regression using convolutional neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 494–495, 2017.
- [MKNT18] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in RGB. *CoRR*, abs/1810.03065, 2018.
- [PLH<sup>+</sup>19] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019.
- [PPV19] Kiru Park, T. Patten, and M. Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7667–7676, 2019.
- [RDGF16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [RL17] 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. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3848–3856, 2017.
- [SMD<sup>+</sup>19] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images, 2019.
- [SSH20] Chen Song, Jiaru Song, and Qixing Huang. Hybridpose: 6d object pose estimation under hybrid representations, 2020.
- [TSF17] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. *CoRR*, abs/1711.08848, 2017.
- [WMS<sup>+</sup>20] Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. Self6d: Self-supervised monocular 6d object pose estimation. In *The European Conference on Computer Vision (ECCV)*, August 2020.
- [XSNF17] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2017.
- [YZKL20] Xin Yu, Zheyu Zhuang, Piotr Koniusz, and Hongdong Li. 6dof object pose estimation via differentiable proxy voting regularizer. In *BMVC*, 2020.
- [ZSI19] Sergey Zakharov, Ivan S. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1941–1950, 2019.