Deep Feedback Inverse Problem Solver

Wei-Chiu Ma[1,2] Shenlong Wang[1,3] Jiayuan Gu[1,4]
Sivabalan Manivasagam[1,3] Antonio Torralba[2] Raquel Urtasun[1,3]

[1]Uber Advanced Technologies Group
[2]Massachusetts Institute of Technology
[3]University of Toronto
[4]University of California San Diego

Abstract. We present an efficient, effective, and generic approach towards solving inverse problems. The key idea is to leverage the feedback signal provided by the forward process and learn an iterative update model. Specifically, at each iteration, the neural network takes the feedback as input and outputs an update on current estimation. Our approach does not have any restrictions on the forward process; it does not require any prior knowledge either. Through the feedback information, our model not only can produce accurate estimations that are coherent to the input observation but also is capable of recovering from early incorrect predictions. We verify the performance of our model over a wide range of inverse problems, including 6-DOF pose estimation, illumination estimation, as well as inverse kinematics. Comparing to traditional optimization-based methods, we can achieve comparable or better performance while being two to three orders of magnitude faster. Compared to deep learning-based approaches, our model consistently improves the performance on all metrics.

1 Introduction

Given a 3D model of an object, the light source(s), and their relevant pose to the camera, one can generate highly realistic images of the scene with one click. While such a *forward* rendering process is complicated and requires explicit modeling of interreflection, self-occlusion, as well as distortion, it is well-defined and can be computed effectively. However, if we were to recover the illumination parameters or predict the 6 DoF pose of the object from the image in an *inverse* fashion, the task becomes extremely challenging. This is because a lot of crucial information is lost during the forward (rendering) process. In fact, many complicated systems in natural science, signal processing, and robotics, all face similar challenges – the model parameters of interest cannot be measured directly and need to be estimated from limited observations. This family of problems are commonly referred to as **inverse problems**. Unfortunately, while there exists sophisticated theories on how to design the forward processes, how to address the inherent ambiguities of the inverse problem still remains an open question.

One popular strategy to disambiguate the problem is to model the inverse problem as a structured optimization task and incorporate human knowledge

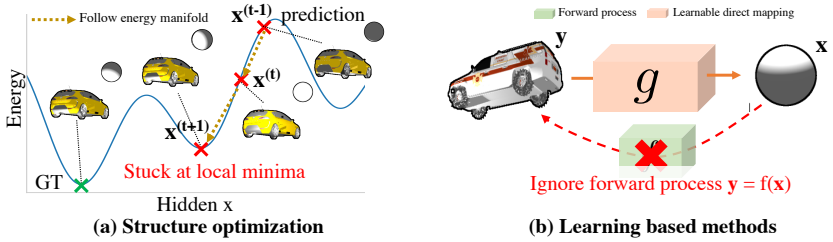
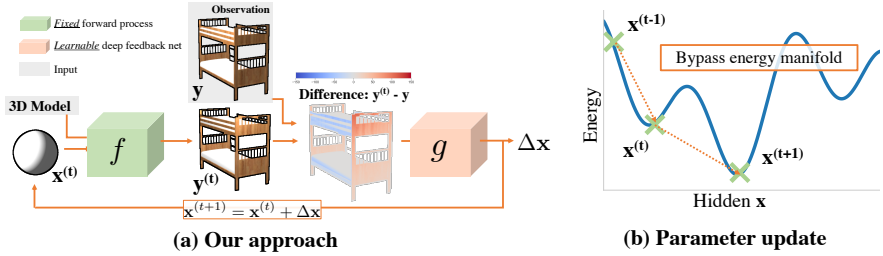


Fig. 1: **Prior work on inverse problems**: (a) Structured optimization approaches require hand-crafted energy/objective functions and are sensitive to initializations which makes them easy to get stuck in local optima. (b) Direct learning based methods do not utilize the available forward process as feedback to guarantee the quality of the solution. Without this feedback, the models cannot rectify the estimates effectively as shown above.

into the model [19,49,21,56]. For instance, the estimated solution should agree with the observation [48] and be smooth [47,3], or should follow a certain statistical distribution [33,64,2]. Through imposing carefully designed objectives, classic structure optimization methods are able to find a solution that not only agrees with the observation but also satisfies our prior knowledge about the solution. In practice, however, almost no hand-crafted priors can succeed in including all phenomena. To ensure that the optimization problem can be solved efficiently, there are multiple restrictions on the form of the priors as well as the the forward process [4], both of which increases the difficulty of design. Furthermore, most optimization approaches require many iterations to converge and are sensitive to initialization.

On the other hand, learning based methods propose to directly learn a mapping from observations to the model parameters [60,24,72,68,55]. They capitalize on powerful machine learning tools to extract task-specific priors in a data-driven fashion. With the help of large-scale datasets and the flourishing of deep learning, they are able to achieve state-of-the-art performance on a variety of inverse problems [69,61,57,13,23]. Unfortunately, these methods often ignore the fact that the forward model for inverse problems is available. Their systems remain open loop and do not have the capability to *update* their prediction based on the *feedback signal*. Consequently, the estimated parameters, while performing well in majority cases, may generate a result that is either incompatible with the real observation or not realistic.

With these challenges in mind, we develop a novel approach to solving inverse problems that takes the best of both worlds. The key idea is to *learn to iteratively update* the current estimation through the *feedback signal* from the forward process. Specifically, we design a neural network that takes the observation and the forward simulation result of the previous estimation as input, and outputs a steepest update towards the ideal model parameters. The advantages are four-fold: First, as each update is trained to aggressively move towards



(a) Our approach **(b) Parameter update**

Fig. 2: **Overview:** Our model iteratively updates the estimation based on the feedback signal from the forward process. We adopt a closed-loop scheme to ensure the consistency between the estimation and the observation. We neither require an objective at test time, nor have any restrictions on the forward process. Click here to watch an animated version of the update procedure.

the ground truth, we can accelerate the update procedure and reach the target with much fewer iterations than classic optimization approaches. Second, our approach does not need to explicitly define the energy. Third, we do not have any restrictions on the forward process, such as differentiability, which greatly expands the applicable domains. Finally, in contrast to the conventional learning methods, our method incorporates feedback signals from the forward process so that the network is aware of how close the current estimation is to the ground truth and can react accordingly. The estimated parameters generally lead to results closer to the observation.

We demonstrate the effectiveness of our approach on three different inverse problems in graphics and robotics: illumination estimation, 6 DoF pose estimation, and inverse kinematics. Compared to traditional optimization based methods, we are able to achieve comparable or better performance while being two to three orders of magnitude faster. Compared to deep learning based approaches, our model consistently improves the performance on all metrics.

2 Background

Let $\mathbf{x} \in \mathcal{X}$ be the hidden parameters of interest and let $\mathbf{y} \in \mathcal{Y}$ be the measurable observations. Denote $f : \mathbf{x} \to \mathbf{y}$ as the deterministic forward process. The aim of inverse problem is to recover $\mathbf{x}$ given the observation $\mathbf{y}$ and the forward mapping f. In the tasks that we consider, $\mathcal{X}$ is a group such as $\mathcal{X} = \mathrm{SE}(3)$ for 6 DoF pose estimation and $\mathcal{X} = \mathbb{R}^3$ when estimating the position of the light source

2.1 Structured optimization

Structured optimization methods generally formulate the inverse problem as an energy minimization task [8,11,12,52,29,48,21]:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} E(\mathbf{x}) = \arg\min_{\mathbf{x}} E_{\text{data}}(f(\mathbf{x}), \mathbf{y}) + \lambda E_{\text{prior}}(\mathbf{x}),$$

Algorithm 1 Deep Feedback Inverse Problem Solver

input observation $\mathbf{y}$, forward model $f(\cdot)$ and init $\mathbf{x}^0$
for $iter = 0, 1, \ldots, T-1$ **do**
    Run forward model: $\mathbf{y}^t = f(\mathbf{x}^t)$
    Compute update: $\mathbf{x}^{t+1} = \mathbf{x}^t + g_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})$
end for
output $\mathbf{x}^T$

where the data term E_{data} measures the similarity between the observation $\mathbf{y}$ and the forward simulated results $f(\mathbf{x})$ of the hidden parameters $\mathbf{x}$; and the prior term E_{prior} encodes humans' knowledge about the solution $\mathbf{x}$. As the energy function is often non-convex, iterative algorithms are used to refine the estimation. Without loss of generality, the update rule can be written as:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + g_E(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y}) \tag{1}$$

where $g_E(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})$ is an analytical update function derived from the energy function E, and $\mathbf{y}^t = f(\mathbf{x}^t)$. For instance, in continuous-valued inverse problems, $g_E = -A_E(\mathbf{x}^t)\nabla_E(\mathbf{x}^t)$, where $\nabla_E(\mathbf{x})$ is the first-order Jacobian and A_E is a warping matrix that depends on the optimization algorithm and the form of the energy. For instance, A_E is simply a (approximated) Hessian matrix in Newton method and is equivalent to the step size in first order gradient descent.

One major advantage of these approaches is that they explicitly take into account how close $f(\mathbf{x})$ and $\mathbf{y}$ are via the data term E_{data}, and exploit such *feedback* as a guidance for the update. This ensures that the result $f(\mathbf{x}^*)$ generated from the final estimation $\mathbf{x}^*$ is close to the observation $\mathbf{y}$. While impressive results have been achieved, there are several challenges remaining: first, they require both the forward process f as well as the prior E_{prior} to be optimization-friendly (*e.g.* differentiable) so that inference algorithms can be applied. Unfortunately this is not the case for many inverse problems and tailored approximations are required [26,42,54,39,67]. The performance may thus be affected. Second, they often require many updates to reach a decent solution (*e.g.* first-order methods). If higher order methods are exploited to speed up the process, the update may become expensive (*e.g.*, second-order methods). Third, carefully designed priors are necessary for identifying the true solution from multiple feasible answers. This is particularly true for ill-posed inverse problems, such as super-resolution and inverse kinematics, in which there exists infinite number of feasible solutions that could generate the observation. Additionally, the energy must be designed in a way that is easy to optimize, which is sometimes non-trivial. Finally, these optimization methods are typically sensitive to the initialization.

2.2 Learning based methods

Another line of work [10,71,32,37,30] has been devoted to directly learning a mapping from the observations $\mathbf{y}$ to the solution $\mathbf{x}$:

$$\mathbf{x}^* = g(\mathbf{y}; \mathbf{w}). \tag{2}$$

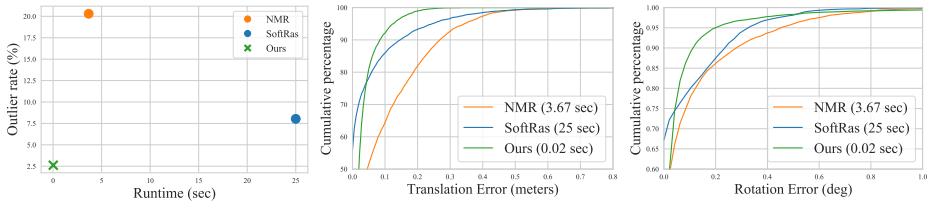


Fig. 3: **Quantitative analysis on 6 DoF pose estimation.** Our deep optimizer is robust, accurate, and significantly faster.

Here, $g(\cdot;\mathbf{w})$ is a learnable function parameterized by $\mathbf{w}$. These approaches try to capitalize on the feature learning capabilities of deep neural networks to extract statistical priors from data, and approximate the inverse process without the help of any hand-crafted energies. While these methods have achieved state-of-the-art performance in many challenging inverse tasks such as inverse kinematics [51,74], super-resolution [32,63], compressive sensing [28], image inpainting [50,41], illumination estimation [35,43], reflection separation [73], and image deblurring [46], they ignore the fact that the forward process f is known.

As a consequence, there is no *feedback* mechanism within the model that scores if $f(\mathbf{x}^*)$ is close to $\mathbf{y}$ after the inference, and the model cannot update the estimation accordingly. The whole system remains *open loop*.

3 Deep Feedback Inverse Problem Solver

In this paper we aim to develop an extremely efficient yet effective approach to solving structured inverse problems. We build our model based on the observation that traditional optimization approaches and current learning based methods are complementary – one is good at exploiting feedback signals as guidance and inducing human priors , while the other excels at learning data-driven inverse mapping. Towards this goal, we present a simple solution that takes the best of both worlds. We first describe our deep feedback network that iteratively updates the solution based on the feedback signal generated by the forward process. Then we demonstrate how to perform efficient inference as well as learning. Finally, we discuss our design choices and the relationships to related work.

3.1 Deep Feedback Network

As we have alluded to above, structured optimization and deep learning have very different yet complementary strengths. Our goal is to bring together the two paradigms, and develop a generic approach to inverse problems.

The key innovation of our approach is to replace the analytical function g_E defined in structured optimization approach at Eq. 1 with a neural network. Specifically, we design a neural network $g_{\mathbf{w}}$ that takes the same set of inputs as g_E and outputs the update. The hope is that the model can perceive the

Methods	Optimization Step	Optimization Time	Trans. Error Mean	Trans. Error Median	Rot. Error Mean	Rot. Error Median	Outlier (%)
NMR [26]	105	3.67 s	0.1	0.05	5.78	1.68	20.3
SoftRas [42]	157	25 s	0.05	0.003	4.14	0.5	8.03
Deep Regression	1	0.004 s	0.07	0.06	10.07	7.68	5
Ours	5	0.02 s	0.02	0.009	2.64	1.02	2.6

Table 1: **Quantitative comparison on 6 DoF pose estimation.**

difference between the observation $\mathbf{y}$ and the simulated forward results $\mathbf{y}^t$ and then predict a new solution based on the *feedback* signal. In practice, we employ a simple addition rule and fold the step size, parameter priors all into $g_{\mathbf{w}}$:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + g_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y}), \quad \text{where } \mathbf{y}^t = f(\mathbf{x}^t). \tag{3}$$

The network architectures design depends on the form of observational data $\mathbf{y}$ and solution $\mathbf{x}$. For instance, for inverse graphics tasks, we utilize convolutional neural networks, since the observations are images. This not only allows us to sidestep all requirements imposed on f (*e.g.* differentiability), but also removes the need for explicitly defining energies. Unlike conventional learning based methods, we take both $\mathbf{y}^t$ and $\mathbf{y}$ as input to the update so that we incorporate the feedback signal through comparing the two.

We derive our final deep structured inverse problem solver by applying the aforementioned update functions in an iterative manner. The algorithm is summarized in Alg. 1. At each step, the solver takes as input the current solution $\mathbf{x}^t$, the observation $\mathbf{y}$, and the forward simulated results $\mathbf{y}^t$, and predicts the next best solution as defined in Eq. 3. In practice, the stopping criteria could either be based on a predefined iteration number or on checking convergence by measuring the difference between solutions from two consecutive iterations.

3.2 Learning

The full deep structured inverse problem solver can be learned in an end-to-end fashion via back-propagation through time (BPTT). Yet in practice we find that applying loss function over each stage's intermediate solution $\mathbf{x}^t$ yields better results. Deep supervision greatly accelerates the speed of convergence.

However, it is non-trivial to design a learning procedure for each iterative update function $g_{\mathbf{w}}$, as there exist infinite paths towards the ideal solution. Ideally, we would like our solution to descend towards the ideal solution as quickly as possible. Thus, inspired by [70], at each iteration, we learn to aggressively predict the update required to reach the ideal solution. At each stage, the learning procedure finds the best $\mathbf{w}$ through minimizing the following loss function:

$$\arg\min_{\mathbf{w}} \sum_{(\mathbf{y}, \mathbf{x}_{\text{gt}})} \sum_{t} \ell(\mathbf{x}_{\text{gt}}, \mathbf{x}^t + g_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})).$$

ℓ is a task-specific loss function; for instance, ℓ is l2-norm for inverse kinematics.

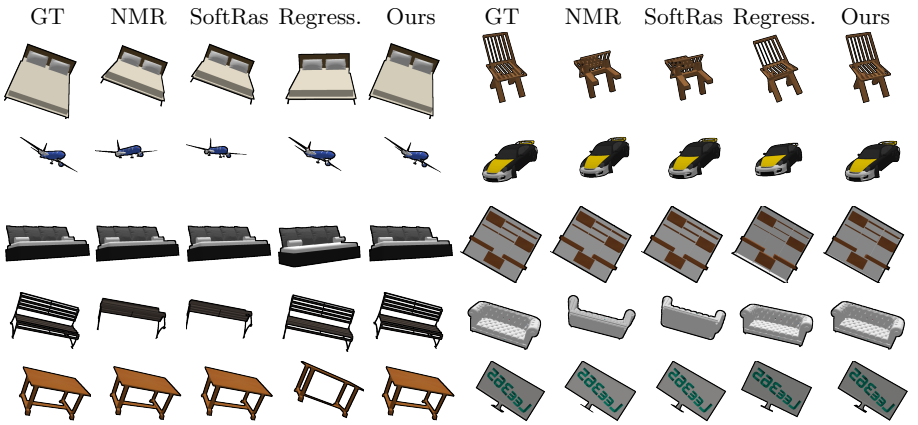


Fig. 4: **Qualitative comparison on 6 DoF pose estimation:** We infer the poses from only silhouette images. The rendered colored images in the figure are for visualization purpose.

3.3 Discussions

Stage-wise network: In our standard approach described before, $g_{\mathbf{w}}$ is shared across all steps. However, the proximity to the ideal solution varies at different step. As a consequence, early iteration often takes inputs that are farther to the ideal solution than what a late iteration update step takes. This brings difficulties to the network as it needs to handle a variety of output scales across different iteration steps. This motivates us to train a separate update function per step $g^t_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})$ that better captures the input data distribution at each iteration. To learn this non-shared weight network, we conduct a stage-wise training procedure. We start to train the $g^0_{\mathbf{w}}$ first. Then acquire $\mathbf{y}^0$ for all the training data, which allow us to train $g^1_{\mathbf{w}}$. We repeat this procedure until $g^T_{\mathbf{w}}$ is trained. In total T models $\{g^t_{\mathbf{w}}\}$ are trained. Please refer to the supp. material for the comparison between sharing weights and not sharing weights.

Adaptive update: Our current update rule is simply an addition, yet it can be easily extended to more sophisticated settings to handle more complex scenarios. For instance, one can apply the classic momentum technique on top of the predicted gradient to stabilize the optimization trajectory. One can also learn another meta-network to dynamically adjust the output of our update network. While all of these options are feasible, we find that in practice a simple strategy suffices. Inspired by the Levenberg-Marquardt method [4], we exploit a damping factor λ to control the effectiveness of the update network, *i.e.*, $\mathbf{x}^{t+1} = \mathbf{x}^t + \lambda \cdot g_{\mathbf{w}}(\mathbf{x}^t, \mathbf{y}^t, \mathbf{y})$. Specifically, λ is initialized to 1 at the beginning of each update. If the new estimation results in a lower data energy than that of the original one, we update the estimation. Otherwise we reduce λ by half

Module	Forward Rendering	Inverse Update	Total
NMR [26]	28 ms	7 ms	35 ms
SoftRas [42]	76 ms	84 ms	160 ms
Ours	2.6 ms	0.9 ms	3.5 ms

Table 2: **Runtime breakdown of a single optimization step for 6 DoF pose estimation.**

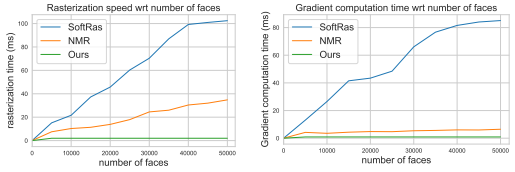


Fig. 5: **Runtime vs number of faces.** (Left) Forward rasterization time. (Right) Backward gradient computation (inverse update) time.

and re-compute. We only need to compute the update gradient once. The forward process is executed on the GPU and hence the computational overhead is negligible. Through this simple rule, we can guarantee that $E_{\text{data}}(\mathbf{x}^t, \mathbf{y})$ decreases after every iteration. Empirically $\mathbf{x}^t$ becomes closer to the ground truth $\mathbf{x}$ as well, since the ambiguity arising from the data term disappears when the estimation is already sufficiently close.

Relationship to existing work: Our model is closely related to the family of iterative networks [17,59,36,5,16,53,58,66,65,7,38,40], in particular the stacked inference machines [53,58,66,65,7]. Unlike previous methods that require the model to implicitly learn the relationship between the input and the preceding estimation, we leverage the forward process to explicitly establish the connection among them and close the loop. This is of crucial importance for inverse problems since the two spaces are very distinct (*e.g.* illumination parameters vs RGB image). The idea of learning to update is inspired by supervised descent methods [70]. However, unlike their approach we learn the mapping and the feature simultaneously. Furthermore, we focus on inverse problems and design a closed-loop scheme to incorporate feedback signals, while they simply perform iterative update in an open loop setting. Developed independently, Flynn *et al.* [14] propose a similar approach for view synthesis. Their model, however, relies on the analytical gradient components. They thus requires the system to be differentiable. In contrast, our approach directly predicts the update from the observation and the feedback signal. We do not require explicit gradient computation and thus do not have such a limitation. Please refer to the supp. material for more discussion on reinforcement learning and other prior art [7,34].

Applicability: Unlike previous work, our approach neither has restrictions on the forward process f, nor need to construct domain-specific objectives at test time. During inference, at each iteration, we simply adopt a feed-forward operation g on top of current estimate and predict the update. Our method is applicable to a wide range of tasks so long as the forward process function f is available. In the following sections, we showcase our approach on two different inverse graphics tasks (object pose estimation and illumination estimation from a single image) as well as one robotics task (inverse kinematics).

Training on $0^\circ - 40^\circ$	Trans. Error		Rot. Error ($^\circ$)	
Evaluation Rot. Range	Mean	Median	Mean	Median
$40^\circ - 45^\circ$	0.05	0.03	11.33	4.97
$45^\circ - 50^\circ$	0.05	0.04	15.62	5.60
$50^\circ - 55^\circ$	0.06	0.04	18.58	6.86
$55^\circ - 60^\circ$	0.07	0.05	24.14	9.58

Table 3: **Test on unseen rotations.**

4 Application I: 6-DoF Object Pose Estimation

Problem formulation: Assume that the 3D model of the object is given [20,6] and the camera intrinsic parameters are known. For a given object pose wrt the camera, denoted as $\mathbf{x} \in \mathrm{SE}(3)$, we can generate the corresponding image observation $\mathbf{y}$ through a forward rendering function $f : \mathbf{x} \rightarrow \mathbf{y}$, powered by a graphics engine. The goal of 6 DoF pose estimation is to *invert* the process and recover the latent pose $\mathbf{x}$ from the observation image $\mathbf{y}$. This problem is particularly important for problems such as robot grasping [34] and self-driving [44]. Unlike previous approaches that leverage RGB information or depth geometry to guide the pose estimation, we focus on a more challenging setting where the observation is *a single silhouette image* $\mathbf{y} \in \{0, 1\}^{H \times W}$. The object pose $\mathbf{x} = (\mathbf{x}_{\text{quat}}; \mathbf{x}_{\text{trans}})$ is represented by a unit quaternion for rotation $\mathbf{x}_{\text{quat}}$ and a 3D translation vector $\mathbf{x}_{\text{trans}}$.

Data: We use the 3D CAD models from ShapeNet [9] within 10 categories: cars, planes, chairs, bench, table, sofa, cabinet, bed, monitor, and couch. The dataset is split into training (70%), validation (10%) and testing (20%). For each object, we randomly sample an axis from the unit sphere and rotate the object around the axis by $\theta \sim [-40, 40]$ degrees. We further translate the object along each axis by a random offset within $[-0.2, 0.2]$ meters. Given the randomly generated ground truth object poses, we render 128×128 silhouette images with non-differentiable PyRender [1] as input observations. We refer the readers to the supp. material for the performance of our model on other image sizes.

Metrics: We measure the translation error with euclidean distance and the rotation error with angular difference. Inspired by [15], we also compute the *outlier ratio* as an indicator of the general quality of the output. Specifically, we define the prediction to be an outlier if the translation error is higher than 0.2 or the rotation error is larger than 30°.

Network architecture: Our deep feedback network $g_{\mathbf{w}}$ is akin to the classic LeNet [31]. It takes as input the rendered image $\mathbf{y}^t = f(\mathbf{x}^t)$, the observed image $\mathbf{y}$, as well as the difference image $\hat{\mathbf{y}} - \mathbf{y}^t$, and directly outputs the update $\Delta\mathbf{x}$. We apply an additional normalization operator over the rotation component to correct it to a valid unit quaternion. We unroll our deep feedback network for

Methods	Optimization Step	Optimization Time	Directional light Mean	Directional light Median	Directional light Outliers	Point light Mean	Point light Median	Point light Outliers
NMR[1] [26]	166.7	58.3 s	0.099	0.037	19.2%	-	-	-
Deep regression [22]	1	0.043 s	0.067	0.022	24%	0.111	0.084	11%
Ours	7	0.183 s	0.052	0.008	8%	0.084	0.064	9%

Table 4: **Illumination estimation on ShapeNet.**

five steps. MSE is employed as the loss function for both rotation and translation since it produces the most stable results.

Baselines: For optimization methods, the energy function consists of a data term $E_{\text{data}}(f(\mathbf{x}), \mathbf{y})$ that favors agreement and a prior term $E_{\text{prior}}(\mathbf{x})$ that encourages the quaternion to remain on the manifold. To make the forward rendering procedure f differentiable, we utilize the state-of-the-art differentiable renderers for comparison, *i.e.* neural mesh renderer (NMR[26]) and soft rasterization (SoftRas [42]). We utilize the following stopping criteria for the optimizer: (i) 500 iterations, or (ii) the silhouette difference between the observation and the one generated by the renderer stops improving for 20 iterations. For the deep regression method, we use the same architecture as our deep feedback network except that no feedback is provided.

Results: As shown in Tab. 1, our method achieves a significantly lower outlier ratio compared to other approaches. This indicates that our model is more robust and less susceptible to becoming stuck in local optimum. It also has comparable performance to differentiable renderers in terms of mean translation and angular error, while being two to three orders of magnitude faster. On the other hand, our method has much better performance than the non-feedback deep regression method. For the category-wise performance, please refer to the supp. material. Fig. 4 showcases some qualitative results. Our method is robust to extreme poses, whereas optimization based method is easy to get stuck in a local optimum.

Deep feedback network as initialization: Due to the highly non-convex structure of the energy model, a good initialization is required for optimization methods to achieve good performance. One natural solution is to exploit our model as an initialization and employ classic solvers for the final optimization. By combining our approach with SoftRas, we can further reduce the error by more than 50%. We refer the readers to supp. material for detailed analysis.

Runtime analysis: We show the runtime break down for a single update step in Tab. 2 and the runtime w.r.t the number of faces in Fig. 5. As we neither need to construct the computation graph nor storing any activation value for

[1] NMR does not support point light. Furthermore, its directional light is highly simplified and did not consider self-occlusion.

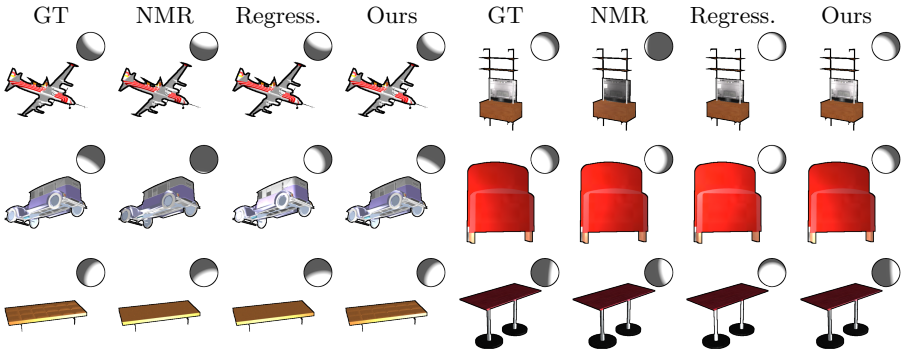


Fig. 6: **Qualitative comparison on illumination estimation.**

gradient computation during the forward rasterization process, our rendering is significantly faster. For gradient computation, SoftRas is far slower as it needs to propagate the gradient to multiple faces. In contrast, our update model is simply an efficient feed-forward neural net that takes as input the (difference) silhouette images. Its speed is invariant to the number of faces.

5 Application II: Illumination Estimation

Problem Formulation: We next evaluate our method on the task of illumination estimation. The goal is to recover the lighting parameter $\mathbf{x} \in \mathbb{R}^3$ from the observation RGB image $\mathbf{y} \in \mathbb{R}^{H \times W \times 3}$. It has critical applications in image relighting and photo-realistic rendering [25]. As in the 6-DoF pose estimation task, we assume the 3D model is given.

Data: We use the same dataset as the 6-DoF pose estimation experiment for the illumination estimation experiment. Specifically, we consider two types of light source: directional light and point light. The two light sources are complementary and can result in very different rendering effect. During training, we randomly sample the light position from the half unit sphere on the camera side [22,43]. If the light is directional, we point the light towards the origin. All the objects are set to have Lambertian surfaces. We ignore the scenario where the light source lies on the other side of the object, as it has no effect on the rendered image. For evaluation, we follow the same criteria. We perform rendering in pyrender and the image size is set to 256×256. Empirically we found this size provides the best balance between performance and the computational speed.

Metrics: Following [22], we use the standard mean-squared error (MSE) between the ground truth light and estimated light pose to measure the difference. We also compute the outlier rate as described in Sec. 4.

	Optimization		Position Error (cm)		Rotation Error (°)	
Methods	Step	Time	Mean	Median	Mean	Median
L-BFGS [18]	73	27.9 s	0.38	0.01	7.19	4.68
Adam [27]	196	38.8 s	0.04	0.04	7.96	7.92
Deep6D [74]	1	0.012 s	1.9	1.6	-	-
Ours	4	0.12 s	0.64	0.36	0.88	0.03

Table 5: **Quantitative results on CMU MoCap.**

Network architecture: We employ an encoder-decoder architecture with skip connections as our deep feedback network. Since the 3D geometry of the object plays an important role during rendering, we adopt depth prediction as an auxiliary task. This allows the model to implicitly capture such notion and reason about its relationship with illumination. During training, our deep feedback network estimates both the depth of the object as well as the illumination parameters. We use MSE as the objective for both tasks. During inference, we simply discard the depth decoder and output only the illumination part. We unroll our network for 7 steps according to the validation performance.

Baselines: We exploit NMR [26] to minimize the energy $E_{\text{data}} + E_{\text{prior}}$.The data term is the ℓ_2 distance between the observation image and the rendered image, while the prior term constrains the light source to lie on the sphere. We adopt the same stopping criteria as in Sec. 4. The size of the rendered image is set to 256×256 based on the performance on the validation set. For deep regression method, we exploit the state-of-the-art model from Janner *et al.* [22].

Results: As shown in Tab. 4, our deep feedback network outperforms the baselines on both setup. The improvement is significant especially in the directional light case. We conjecture this is because the intensity of directional light does not decay w.r.t. the travel distance, and the signals from the image are weaker. Learning based approaches thus have to rely on feedback signals to refine the light direction. The performance of the optimization method is limited by the hand-crafted energy as well as the capability of renderer. NMR is sub-optimal as it approximates the gradient with a manually designed function and does not handle self-occlusion. In contrast, our method allows us to exploit complex rendering machines as the forward model as we do not require it to be differentiable. We note that we only report the optimization results on directional light since NMR does not support point light source. Fig. 6 depicts the qualitative comparison against the baselines. It is clear that our deep feedback mechanism is able to recover accurate lighting information based on subtle difference between the forward results and the observations.

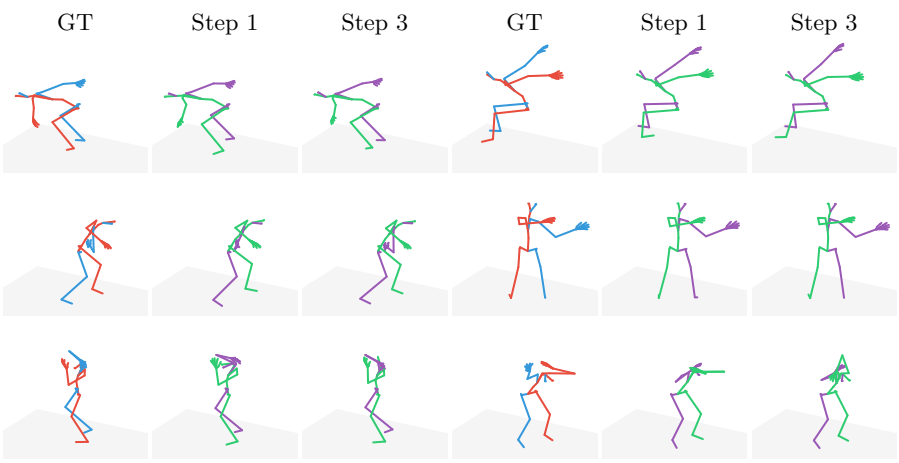


Fig. 7: **Qualitative results on CMU MoCap:** Our approach is able to accurately predict the joint rotations within a few steps. It can also correct wrong estimations through the feedback from the forward model (see the feet/toes in the right column). Bottom right shows an example where our model fails.

6 Application III: Inverse Kinematics

Problem formulation Finally we exploit how our proposed method to tackle the inverse kinematics problem. Given the 3D location of the joints of a reference pose $\mathbf{y}^{\text{ref}}_{1:N}$ and the desired joint rotations $\mathbf{x}_{1:N} \in \text{SO}(3)$, the forward kinematics function f rotates the joints and computes their 3D positions by recursively applying the follow update rule from parents to children: $\mathbf{y}_n = \mathbf{y}_{\text{parent}(n)} + \mathbf{x}_n(\mathbf{y}^{\text{ref}}_n - \mathbf{y}^{\text{ref}}_{\text{parent}(n)})$. The goal of inverse kinematics is to recover the SO(3) rotations $\mathbf{x}_{1:N}$ that ensure the specific joints are placed at the desired 3D locations $\mathbf{y}_{1:N}$. Inverse kinematics has a wide range of applications, such as robot arm manipulation, legged robot motion planning and computer re-animation. The problem is inherently ill-posed as different rotations can result in the same observation through the forward kinematics function f, *i.e.*, $\mathbf{y} = f(\mathbf{x}_{1:N}) = f(\mathbf{x}'_{1:N})$. However, not all angles are feasible or natural due to the dynamic constraints. Therefore, in order to accurately recover the rotations, one has to either come up with a powerful prior or learn it from data. In this paper, we focus on inverse kinematics over human body skeletons.

Data: We validate our model on the CMU Motion Capture Dataset (CMU MoCap) as it contains complex human motions and a diverse range of joint rotations. Following Yi *et al.* [74], we select 865 motion clips from 37 motion categories and hold out 37 clips for testing. Each skeleton in the dataset has 57 joints. We fix the position of the hip to remove the effect of global motion.

Metrics: We evaluate the performance of our model with joint position error [74] and joint angular error [45,51]. The two metrics are complementary since a small rotation error may result in a large position error due to the recursive nature of the forward kinematics model, and small position error cannot guarantee correct joint rotation due to ambiguities.

Network architecture: Our deep feedback network is a multilayer perception akin to [74]. Following [62,51], the network takes as input the estimated joint position, reference joint position, as well as the difference between the two, and outputs a rotation for each joint. We unroll our model three steps. We train the network with L2 loss on both position error and rotation error.

Baselines: We compare our model against two optimization-based approaches and one deep regression method. For optimization methods, we employ joint position error as our data term, *i.e.* $E_{\text{data}}(f(\mathbf{x}), \mathbf{y}) = \|f(\mathbf{x}) - \mathbf{y}\|_2^2$, and derive a prior energy term from data to alleviate the ambiguities of joint rotations. In particular, we fit a gaussian distribution over the Euler angles of each joint from training data and employ it as a regularization term during inference. We set the weight of the prior term to 0.001 and optimize both energies jointly. We exploit two different types of optimizers: a first-order method (*i.e.*, Adam [27]) and a quasi-Newton method (*i.e.*, L-BFGS [18]). For deep regression method, we compare with the current state of the art (Deep6D [74]).

Results: As shown in Tab. 5, our deep feedback network outperforms the baselines on the rotation metric and achieve comparable performance on the position error. By unrolling more steps and gathering feedback signals from the forward model, we are able to reduce incorrect estimation and improve the performance (see the Fig. 7). We refer the readers to the supp. material for detailed analysis. On average, a single step of L-BFGS, Adam, and our approach takes 383 ms, 198 ms, 30 ms respectively. L-BFGS takes longer to compute as it needs to conduct gradient evaluation multiple times to approximate the Hessian. Adam is faster in terms of computation, yet it takes far more steps to converge. Our approach, in comparison, is significantly faster and better.

7 Conclusions

In this paper, we propose a deep feedback inverse problem solver. Our method combines the strength of both learning-based approaches and optimization-based methods. Specifically, it learns to conduct an iterative update over the current solution based on the feedback signals provided from the forward process of the problem. Unlike prior work, it does not have any restrictions on the forward process. Further, it learns to conduct an update without explicitly define an objective function. Our results showcase that the proposed method is extremely effective, efficient, and widely applicable.

References

1. Pyrender. https://github.com/mmatl/pyrender (2020) 9
2. Barron, J.T., Malik, J.: Shape, illumination, and reflectance from shading. TPAMI (2014) 2
3. Bell, S., Bala, K., Snavely, N.: Intrinsic images in the wild. TOG (2014) 2
4. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge university press (2004) 2, 7
5. Byeon, W., Breuel, T.M., Raue, F., Liwicki, M.: Scene labeling with lstm recurrent neural networks. In: CVPR (2015) 8
6. Cao, Z., Sheikh, Y., Banerjee, N.K.: Real-time scalable 6dof pose estimation for textureless objects. In: ICRA (2016) 9
7. Carreira, J., Agrawal, P., Fragkiadaki, K., Malik, J.: Human pose estimation with iterative error feedback. In: CVPR (2016) 8
8. Chan, T.F., Shen, J., Zhou, H.M.: Total variation wavelet inpainting. Journal of Mathematical imaging and Vision (2006) 3
9. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv (2015) 9
10. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: ECCV (2014) 4
11. Dong, W., Zhang, L., Shi, G., Wu, X.: Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. TIP (2011) 3
12. Donoho, D.L.: De-noising by soft-thresholding. Transactions on Information Theory (1995) 3
13. Epstein, D., Chen, B., Vondrick, C.: Oops! predicting unintentional action in video. arXiv (2019) 2
14. Flynn, J., Broxton, M., Debevec, P., DuVall, M., Fyffe, G., Overbeck, R., Snavely, N., Tucker, R.: Deepview: View synthesis with learned gradient descent. arXiv (2019) 8
15. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012) 9
16. Gkioxari, G., Toshev, A., Jaitly, N.: Chained predictions using convolutional neural networks. In: ECCV (2016) 8
17. Greff, K., Srivastava, R.K., Schmidhuber, J.: Highway and residual networks learn unrolled iterative estimation. arXiv (2016) 8
18. Grochow, K., Martin, S.L., Hertzmann, A., Popović, Z.: Style-based inverse kinematics. In: TOG (2004) 12, 14
19. He, K., Sun, J., Tang, X.: Single image haze removal using dark channel prior. TPAMI (2010) 2
20. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: ACCV (2012) 9
21. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: CVPR (2015) 2, 3
22. Janner, M., Wu, J., Kulkarni, T.D., Yildirim, I., Tenenbaum, J.: Self-supervised intrinsic image decomposition. In: NeurIPS (2017) 10, 11, 12
23. Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: CVPR (2018) 2

24. Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections. In: ECCV (2018) 2
25. Karsch, K., Hedau, V., Forsyth, D., Hoiem, D.: Rendering synthetic objects into legacy photographs. TOG (2011) 11
26. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: CVPR (2018) 4, 6, 8, 10, 12
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv (2014) 12, 14
28. Kulkarni, K., Lohit, S., Turaga, P., Kerviche, R., Ashok, A.: Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In: CVPR (2016) 5
29. Laffont, P.Y., Bazin, J.C.: Intrinsic decomposition of image sequences from local temporal variations. In: ICCV (2015) 3
30. Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H.: Deep laplacian pyramid networks for fast and accurate super-resolution. In: CVPR (2017) 4
31. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998) 9
32. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR (2017) 4, 5
33. Levin, A.: Blind motion deblurring using image statistics. In: NeurIPS (2007) 2
34. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: ECCV (2018) 8, 9
35. Li, Z., Snavely, N.: Learning intrinsic image decomposition from watching the world. In: CVPR (2018) 5
36. Liang, M., Hu, X.: Recurrent convolutional neural network for object recognition. In: CVPR (2015) 8
37. Lin, C.H., Kong, C., Lucey, S.: Learning efficient point cloud generation for dense 3d object reconstruction. In: AAAI (2018) 4
38. Lin, C.H., Lucey, S.: Inverse compositional spatial transformer networks. In: CVPR (2017) 8
39. Lin, C.H., Wang, O., Russell, B.C., Shechtman, E., Kim, V.G., Fisher, M., Lucey, S.: Photometric mesh optimization for video-aligned 3d object reconstruction. In: CVPR (2019) 4
40. Lin, C.H., Yumer, E., Wang, O., Shechtman, E., Lucey, S.: St-gan: Spatial transformer generative adversarial networks for image compositing. In: CVPR (2018) 8
41. Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: ECCV (2018) 5
42. Liu, S., Chen, W., Li, T., Li, H.: Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. arXiv (2019) 4, 6, 8, 10
43. Ma, W.C., Chu, H., Zhou, B., Urtasun, R., Torralba, A.: Single image intrinsic decomposition without a single intrinsic image. In: ECCV (2018) 5, 11
44. Ma, W.C., Wang, S., Hu, R., Xiong, Y., Urtasun, R.: Deep rigid instance scene flow. In: CVPR (2019) 9
45. Martinez, J., Black, M.J., Romero, J.: On human motion prediction using recurrent neural networks. In: CVPR (2017) 14
46. Nah, S., Hyun Kim, T., Mu Lee, K.: Deep multi-scale convolutional neural network for dynamic scene deblurring. In: CVPR (2017) 5
47. Oh, B.M., Chen, M., Dorsey, J., Durand, F.: Image-based modeling and photo editing. In: SIGGRAPH (2001) 2

48. Pan, J., Hu, Z., Su, Z., Yang, M.H.: l_0-regularized intensity and gradient prior for deblurring text images and beyond. TPAMI (2016) 2, 3
49. Pan, J., Sun, D., Pfister, H., Yang, M.H.: Blind image deblurring using dark channel prior. In: CVPR (2016) 2
50. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: CVPR (2016) 5
51. Pavllo, D., Grangier, D., Auli, M.: Quaternet: A quaternion-based recurrent model for human motion. In: BMVS (2018) 5, 14
52. Portilla, J., Strela, V., Wainwright, M.J., Simoncelli, E.P.: Image denoising using scale mixtures of gaussians in the wavelet domain. TIP (2003) 3
53. Ramakrishna, V., Munoz, D., Hebert, M., Bagnell, J.A., Sheikh, Y.: Pose machines: Articulated pose estimation via inference machines. In: ECCV (2014) 8
54. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.Y., Johnson, J., Gkioxari, G.: Pytorch3d. https://github.com/facebookresearch/pytorch3d (2020) 4
55. Rick Chang, J., Li, C.L., Poczos, B., Vijaya Kumar, B., Sankaranarayanan, A.C.: One network to solve them all–solving linear inverse problems using deep projection models. In: ICCV (2017) 2
56. Rother, C., Kiefel, M., Zhang, L., Schölkopf, B., Gehler, P.V.: Recovering intrinsic images with a global sparsity prior on reflectance. In: NeurIPS (2011) 2
57. Shocher, A., Cohen, N., Irani, M.: "zero-shot" super-resolution using deep internal learning. In: CVPR (2018) 2
58. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: CVPR (2014) 8
59. Tu, Z.: Auto-context and its application to high-level vision tasks. In: CVPR (2008) 8
60. Tung, H.Y., Tung, H.W., Yumer, E., Fragkiadaki, K.: Self-supervised learning of motion capture. In: NeurIPS (2017) 2
61. Tung, H.Y.F., Harley, A.W., Seto, W., Fragkiadaki, K.: Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In: ICCV (2017) 2
62. Villegas, R., Yang, J., Ceylan, D., Lee, H.: Neural kinematic networks for unsupervised motion retargetting. In: CVPR (2018) 14
63. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., Change Loy, C.: Esrgan: Enhanced super-resolution generative adversarial networks. In: ECCV (2018) 5
64. Wang, Z., Yang, Y., Wang, Z., Chang, S., Yang, J., Huang, T.S.: Learning super-resolution jointly from external and internal examples. TIP (2015) 2
65. Wei, S.E., Ramakrishna, V., Kanade, T., Sheikh, Y.: Convolutional pose machines. In: CVPR (2016) 8
66. Weiss, D., Taskar, B.: Structured prediction cascades. In: AISTATS (2010) 8
67. Wiles, O., Gkioxari, G., Szeliski, R., Johnson, J.: Synsin: End-to-end view synthesis from a single image. arXiv (2019) 4
68. Wu, J., Lim, J.J., Zhang, H., Tenenbaum, J.B.: Physics 101: Learning physical object properties from unlabeled videos. 2
69. Wu, J., Yildirim, I., Lim, J.J., Freeman, B., Tenenbaum, J.: Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In: NeurIPS (2015) 2
70. Xiong, X., De la Torre, F.: Supervised descent method and its applications to face alignment. In: CVPR (2013) 6, 8
71. Xu, L., Ren, J.S., Liu, C., Jia, J.: Deep convolutional neural network for image deconvolution. In: NeurIPS (2014) 4

72. Yao, S., Hsu, T.M., Zhu, J.Y., Wu, J., Torralba, A., Freeman, B., Tenenbaum, J.: 3d-aware scene manipulation via inverse graphics. In: NeurIPS (2018) 2
73. Zhang, X., Ng, R., Chen, Q.: Single image reflection separation with perceptual losses. In: CVPR (2018) 5
74. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: CVPR (2019) 5, 12, 13, 14