Supplementary Material: Deep Feedback Inverse Problem Solver

Wei-Chiu Ma[1,2] Shenlong Wang[1,3] Jiayuan Gu[1,4] Siva 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. This supplementary material provides more details and thorough analysis of our deep feedback inverse problem solver. We hope the readers can gain more insights into our approach. In the following, we first showcase that our model can be combined with traditional optimizers to further improve the performance. Next, we provide more detailed ablation studies regarding the design choices that we made. Then we discuss in depth the relationships between our work and several previous approaches. To demonstrate that our method is not limited to the tasks shown in the paper, we further apply it to a dense inverse image reconstruction problem — JPEG image deblocking. Finally, we present more qualitative results.

1 Analysis

Deep feedback network as initialization: Structure optimization approaches are usually sensitive to initialization. The initial estimations have to be reasonably close so that the optimizers can converge to reasonably good results. Since our deep feedback network can bypass the local energy landscape and can produce very accurate estimations within extremely short amount of time, one natural solution is to exploit our model as an initialization and employ classic solvers for the final optimization. As shown in Tab. 1, by combining with SoftRas [4], we can reduce the error by a significant margin. To further understand how robust the joint model is and how often it can converge to a certain error range, we visualize the cumulative error in Fig. 1. The joint model significantly outperforms all approaches at 90^{th}, 95^{th}, and even 99^{th} percentile, verifying our hypothesis that our deep feedback network is more robust to the initialization as well as the curvature of the local energy landscape, and can serve as a good initializer.

Effectiveness of unrolling steps: Tab. 2 shows the step by step performance of our model on three different tasks. Through iterative feedback and update, our deep feedback inverse problem solver can significantly reduce the overall error. We also provide several side-by-side visualizations in the attached `gif` files. We encourage the readers to watch the animations to see how the estimations evolve.

Methods	Trans. Error Mean	Trans. Error Median	Rot. Error Mean	Rot. Error Median	Outlier (%)
NMR [2]	0.1	0.05	5.78	1.68	20.3
SoftRas [4]	0.05	0.003	4.14	0.5	8.03
Deep Regression	0.07	0.06	10.07	7.68	5
Ours	0.02	0.009	2.64	1.02	2.6
Ours + SoftRas	**0.01**	**0.001**	**1.62**	**0.31**	**0.9**

Table 1: **Deep feedback inverse problem solver as initializer.**

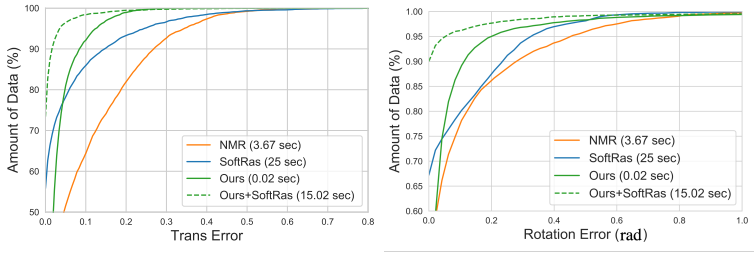

Fig. 1: **Cumulative error for translation and rotation.**

Shared weight vs non-shared weight: In the original paper, we argue that a non-shared weight network can model the dynamic output scale much easily. To corroborate our conjecture, we train a model following the exact same setting except the weight-sharing scheme. As shown in Tab. 3, the non-shared weight network has a larger capacity and can better capture the dynamic distribution.

Category-wise performance: As shown in Tab. 4, our model achieves similar performance across all categories except `bed` and `cabinet`. The two categories are extremely challenging because of the large intra-class variance. For instance, while `hammocks`, `loft bed`, and `ordinary bed` all belong to `bed` category, their shape and appearance vary significantly. In contrast, the shape variations among all cars are smaller and thus our model can pick up such shape prior much easily.

Influence of image size: Ideally we want to provide our model with as much information as possible. One straightforward solution is to exploit a larger rendered image. Unlike state-of-the-art differentiable renderers [2,4] whose runtime increases significantly with image sizes, the computational overhead of pyrender is almost negligible (see Fig. 2). This allows us to freely select the desired image

	Unroll Steps		
6 DoF Pose Est.	1	3	5
Trans. error	0.075	0.032	0.02
Rot. error	10.07	3.32	2.64

	Unroll Steps		
Illum. Est.	1	3	7
Dir. light	0.076	0.055	0.052
Point light	0.111	0.089	0.084

	Unroll Steps		
Inv. kinematics	1	2	3
Position error	2.76	1.11	0.64
Rotation error	2.04	0.94	0.88

Table 2: **Effectiveness of unrolling steps.**

Tasks	6 DoF Pose Est.		Illumination Est.		Inv. Kinematics	
Our model	Translation	Rotation	Directional	Point	Position	Rotation
Shared weight	0.03	4.24	0.077	0.103	0.99	1.57
Non-shared weight	**0.02**	**2.64**	**0.052**	**0.085**	**0.64**	**0.88**

Table 3: **Effectiveness of not sharing weight across stages.**

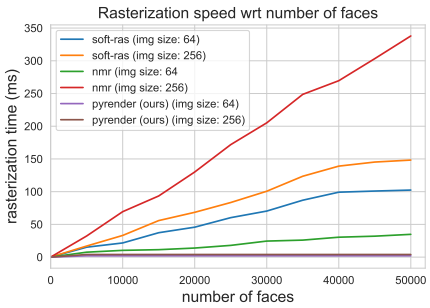

Fig. 2: **Runtime w.r.t. number of faces and image size.**

size without sacrificing the speed. Tab. 4 shows the results of our model with different input image sizes. The performance improves across all categories when the input image size increases from 64 to 128. Yet the results remains roughly the same when increasing the size from 128 to 256. Considering the slightly higher memory cost and slower speed, we select 128 as our final rendered image size.

2 Related work

Connection to reinforcement learning (RL): Our method shares similarities with RL. Indeed, both frameworks are closed-loop systems, where the model takes the feedback from the "environment" and adjusts the next esti-

	Optimization	Couch		Car		Bench		Monitor		Chair	
Image Size	Runtime	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.
64	16 ms	0.02	1.48	0.02	1.33	0.02	1.98	0.02	2.85	0.02	1.80
128	21 ms	**0.009**	**0.81**	**0.008**	**0.60**	**0.009**	1.10	**0.01**	1.49	**0.01**	0.95
256	35 ms	0.01	1.22	0.009	0.67	0.01	**1.05**	**0.01**	**1.33**	**0.01**	**0.92**

	Memory	Table		Sofa		Plane		Bed		Cabinet	
Image Size	Usage	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.
64	583 MB	0.02	2.76	0.02	1.61	0.08	3.83	0.08	7.78	0.05	5.83
128	636 MB	**0.01**	1.28	**0.009**	**0.85**	0.05	1.98	**0.07**	**4.56**	**0.02**	3.69
256	712 MB	**0.01**	**1.18**	0.01	0.87	**0.02**	**1.14**	0.08	4.88	**0.02**	**3.49**

Table 4: **Effectiveness of image size on 6 DoF pose estimation.**

mation accordingly. There exists, however, several key differences: First, the training strategy is different. While our approach exploits GT $\mathbf{x}_{\text{gt}}$ to directly train the feedback network $g_{\mathbf{w}}$ in a supervised fashion, RL agents learn by interacting with the closed-loop environment. The training signals of RL come from non-differentiable rewards $\mathbf{y} - \mathbf{y}^t$. Second, our model is trained to aggressively move towards the ground truth at each step. Thus we can accelerate the update procedure and reach the target with much fewer iterations. In comparison, RL agents are usually restricted to relatively small action space, due to the sampling efficiency and exploration issue, and typically require many more steps to arrive at the final solution. While one can augment the action space, it may bring difficulties to train RL agents in a sample efficient manner.

Comparison with DeepIM [3]: In this paper, we present a *generic framework* that is applicable to a wide range of inverse problems. While the instantiation of our model on 6D pose estimation is similar to the method Li *et al.* introduced in [3], there are a few key differences: (1) Li *et al.* *implicitly* model the relationships between the estimation and the observation. In contrast, we *explicitly* consider the difference and predict an update based upon it. Empirically we find that the explicit representation is crucial for learning and can drastically reduce the size of the model. (2) Our model infers the 6d pose merely from silhouette images, yet [3] focuses on RGB images. (3) Our model is motivated by classic optimization approaches. We borrow ideas from traditional literature to improve the performance (*i.e.* adaptive update), whereas [3] simply unroll the network.

Comparison with IEF [1]: Our approach is related to [1]. Both work leverage feedback signals to refine the estimation iteratively. However, instead of relying on the network to implicitly establish the relationships between the feedback signal and the observation, we explicitly leverage the forward process to map the estimation back to the observation space and compare the difference. We empirically find that ensuring the inputs to lie in the same space is crucial for learning and can drastically reduce the size of the model. Moreover, unlike [1], our predicted update is not bounded. At each iteration, we aggressively move towards the GT solution and hence we can converge within a few iterations. While [1] argues that unbounded update is difficult, we find it possible thanks to the rich information encoded in the difference image. Finally, we focus on a wide range of inverse problems, whereas [1] is specialized to human pose estimation.

Comparison with Oberweger *et al.* [5]: While both work attempt to train networks that take feedback signals as input, there are a few key differences: (1) Instead of relying on the network to *implicitly* establish the relationships between the feedback signal and the observation, we *explicitly* consider the difference and predict an update based upon it. (2) Rather than reusing the same update network for each iteration (*i.e.*, sharing weight), we train a separate update function at different steps. This allows us to handle a variety of output

scales across different iteration steps. (3) We directly supervise our feedback network to move aggressively towards the ideal solution at each stage. In contrast, Oberweger *et al*. [5] encourages the update network to improve the estimation, no matter what scale it is. (4) Oberweger *et al*. [5] learn the synthesis function and focus on hand pose estimation, whereas we target a wide range of inverse problems where the forward process f is given.

Comparison with LiDO [6]: The two papers indeed share many similarities despite developed independently. Both papers are motivated by energy-based models; both discover that the difference image contains very rich information; both learn a deep net to update the latent parameters. There, however, still exists a few differences: (1) Rather than directly regressing the GT [6], we predict the residual instead. (2) LiDO's [6] prediction networks are the same across all iterations, while our feedback network differs at each step. Furthermore, we adopt a stage-wise training to explicitly condition the latter feedback network on the previous ones to more effectively model the variety of output scales.

3 Other applications: JPEG image deblocking

Our approach is not limited to the tasks shown in the paper. It is designed in a generic fashion such that it can be applied to various inverse problems so long as the corresponding forward process f is given. This includes inverse image reconstruction problems such as inverse halftoning, JPEG compression noise removal, super resolution, etc. The underlying training and inference procedures for these tasks are the same as in the paper with the latent variable $\mathbf{x}$ now being high-dimensional (i.e., images).

To verify our claim, we test our approach on a classic inverse image processing task - JPEG image deblocking. Let $\mathbf{x}$ be a clean image, and $\mathbf{y} = f(\mathbf{x})$ be the compressed JPEG image generated by the forward non-differentiable JPEG compressor f. Our goal is to learn an inverse network g that can restore the original image $\mathbf{x} = g(\mathbf{y})$ from the low-quality observation.

We follow a similar experimental setup to [8], where we train our model on BSD400 and evaluate it on BSD68 as well as Live29. We compare our method against state-of-the-art DnCNN [8] as well as Deep Image Prior (DIP) [7]. Our approach improves DIP by 2.33/4.15 db in PSNR and by 0.1076/0.1851 in SSIM (BSD68/Live29). Compared to DnCNN, we improve the PSNR by 0.04/0.03 db and SSIM by 0.001/0.001. We note that the above results are obtained by simple plug-and-play, without hyper-parameter tuning. We unroll our model 3 times.

4 Qualitative results

We provide more qualitative results of our model in Fig. 3 and Fig. 4. Through iterative feedback and update, our method is able to accurately infer the hidden parameters of interest. Furthermore, it can even recover from incorrect predictions in early stages. For instance, the initial pose estimations of the gray car and

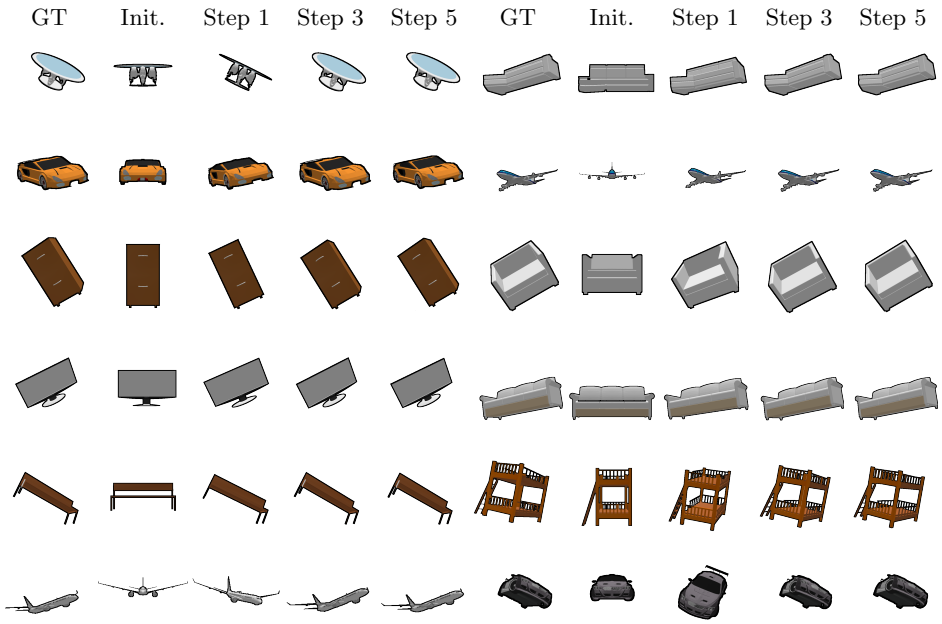

Fig. 3: **Qualitative results on 6 DoF pose estimation:** Our model can accurately estimate the 6 DoF pose of the object from a variety of viewpoints. It can also recover from incorrect estimation through iterative feedback. The images are rendered with the estimated/gt pose, purely for visualization purpose. We only exploit silhouette both during training and inference.

the air plane (last row in Fig. 3) are completely wrong. Yet with the feedback signal, the model is able to iteratively refine the estimation and finally produce decent results.

References

1. Carreira, J., Agrawal, P., Fragkiadaki, K., Malik, J.: Human pose estimation with iterative error feedback. In: CVPR (2016) 4
2. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: CVPR (2018) 2
3. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: ECCV (2018) 4
4. Liu, S., Chen, W., Li, T., Li, H.: Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. arXiv (2019) 1, 2
5. Oberweger, M., Wohlhart, P., Lepetit, V.: Training a feedback loop for hand pose estimation. In: ICCV (2015) 4, 5
6. Romaszko, L., Williams, C.K., Winn, J.: Learning direct optimization for scene understanding. Pattern Recognition (2020) 5
7. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. In: CVPR (2018) 5

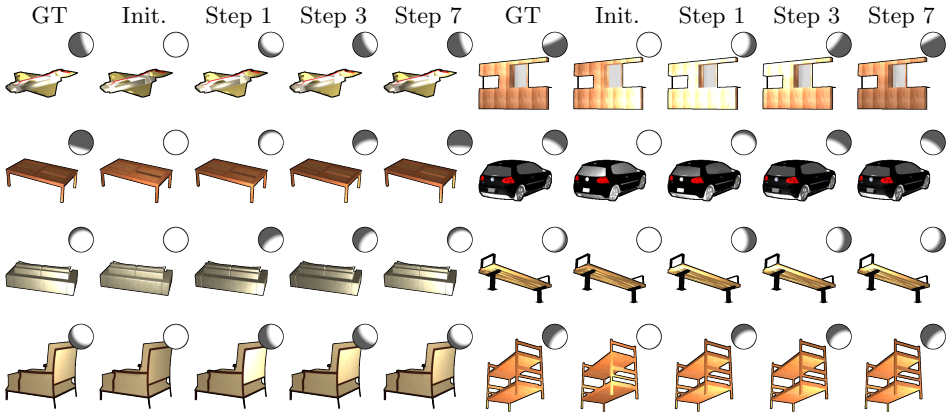

Fig. 4: **Qualitative results on illumination estimation.** While the initial estimations are not that accurate, our model is able to aggressively refine the prediction based on the feedback signal and achieve decent results. The input lights are visualized by rendering them onto a sphere (top-right).

8. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. TIP (2017) 5