Supplementary Material: Proximal Deep Structured Models

Shenlong Wang
University of Toronto
slwang@cs.toronto.edu

Sanja Fidler
University of Toronto
fidler@cs.toronto.edu

Raquel Urtasun
University of Toronto
urtasun@cs.toronto.edu

Abstract

In this supplementary material we first show the analogy between other proximal methods and our proposed deep structured model, including proximal gradient method and alternating direction method of multipliers. After that, we provide more quantitive results on the three experiments.

1 More Proximal Algorithms Examples

Let us the consider the problem we defined in Eq. 1 in our main submission. We aim at tackling the following inference problem:

$$\mathbf{y}^* = \arg\min_{\mathbf{y}\in\mathcal{Y}} E(\mathbf{x},\mathbf{y};\mathbf{w}) = \arg\min_{\mathbf{y}\in\mathcal{Y}} \sum_i f_i(y_i,\mathbf{x};\mathbf{w}_{\mathrm{u}}) + \sum_\alpha f_\alpha(\mathbf{y}_\alpha,\mathbf{x};\mathbf{w}_\alpha) \tag{1}$$

where $f_i(y_i;\mathbf{x},\mathbf{w}) : \mathcal{Y}_i \times \mathcal{X} \to \mathbb{R}$ is a function that depends on a single variable (i.e., a unary term) and $f_\alpha(\mathbf{y}_\alpha) : \mathcal{Y}_\alpha \times \mathcal{X} \to \mathbb{R}$ depends on a subset of variables $\mathbf{y}_\alpha = (y_i)_{i\in\alpha}$ defined on a domain $\mathcal{Y}_\alpha \subset \mathcal{Y}$.

We focus on the problem with potential functions satisfying the following conditions:

1. There exists a function h_i and g_i such that $f_i(y_i,\mathbf{x};\mathbf{w}) = g_i(y_i, h_i(\mathbf{x},\mathbf{w}))$, where g_i is a distance function;
2. There exists a closed-form proximal operator for $g_i(y_i, h_i(\mathbf{x};\mathbf{w}))$ wrt y_i;
3. There exist functions h_α and g_α such that $f_\alpha(y_\alpha,\mathbf{x};\mathbf{w})$ can be re-written as $f_\alpha(y_\alpha,\mathbf{x};\mathbf{w}) = h_\alpha(\mathbf{x};\mathbf{w})g_\alpha(\mathbf{w}_\alpha^T\mathbf{y}_\alpha)$;
4. There exists a proximal operator for either the dual or primal form of $g_\alpha(\cdot)$.

1.1 Proximal Gradient Method

Given that all the conditions (1-3) hold true and g_α is differentiable wrt $\mathbf{y}$, we are able to apply proximal gradient method to tackle Eq. (1). We denote $\sum_i f_i(y_i,\mathbf{x};\mathbf{w}_{\mathrm{u}})$ as $E_{\mathrm{u}}(\mathbf{y})$ and $\sum_\alpha f_\alpha(\mathbf{y}_\alpha,\mathbf{x};\mathbf{w}_\alpha)$ as $E_{\mathrm{ho}}(\mathbf{y})$ for simplicity. If g_α is differentiable wrt $\mathbf{y}$, then $E_{\mathrm{ho}}(\mathbf{y})$ is differentiable We can then utilize proximal gradient descent to tackle the inference problem. The general idea of the proximal gradient descent method is based on the fix-point theory: a point $\mathbf{y}^*$ is a solution that minimizes (1) if and only if

$$\mathbf{0} \in \nabla E_{\mathrm{ho}}(\mathbf{y}) + \partial E_{\mathrm{u}}(\mathbf{y})$$

where $\nabla E_{\mathrm{ho}}(\mathbf{y})$ is the gradient of $E_{\mathrm{ho}}(\mathbf{y})$ and $\delta E_{\mathrm{u}}(\mathbf{y})$ is the subgradient for $E_{\mathrm{u}}(\mathbf{y})$. Motivated by this we can derive the following updating rule:

$$\mathbf{y}^{(t+1)} = \mathrm{prox}_{E_{\mathrm{u}}}(\mathbf{y}^{(t)} - \sigma_u \nabla E_{\mathrm{ho}}(\mathbf{y}))$$

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

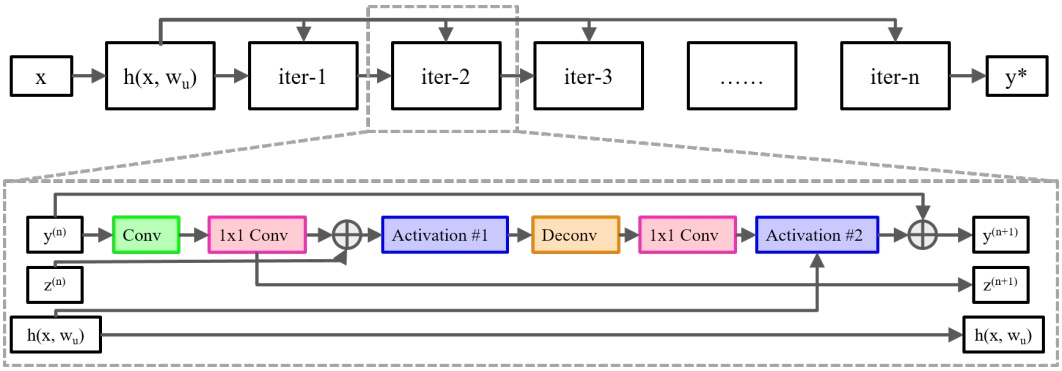

Figure 1: The whole architecture (top) and one iteration block (bottom) of the proposed deep structured network.

According to the chain rule and condition (3) we have $\nabla E_{\text{ho}}(\mathbf{y}) = \sum_\alpha h(\mathbf{x}, \mathbf{w}) \nabla g_\alpha(\mathbf{w}^T \mathbf{y}) \mathbf{w}_\alpha$. We can thus derive the following update rule based on the chain rule in gradient computation:

$$\begin{cases} z_\alpha^{(t+1)} &= \nabla g_\alpha(0 + 1 \cdot \mathbf{w}_\alpha^T \mathbf{y}_\alpha^{(t)}) \\ y_i^{(t+1)} &= \text{prox}_{g_i, h_i(\mathbf{x}, \mathbf{w})}(y_i^{(t)} - \frac{\sigma_u}{h_\alpha(\mathbf{x}; \mathbf{w})} \mathbf{w}_{\cdot, i}^{*T} \mathbf{z}^{(t+1)}) \\ \bar{y}^{(n+1)} &= y^{(n+1)} + \sigma_{ex}(y^{(n+1)} - y^{(n)}) \end{cases} \tag{2}$$

where σ_u is the hyperparameter that represents the gradient step for high-order term and σ_{ex} a hyperparameter that represents the extrapolation gradient step. If the extrapolation gradient step equals to 0, the abo ve update rule corresponds to the standard proximal gradient algorithm. From this set of equations we can see that each step of the iteration corresponds to our building block of the structured neural network in Fig. 1. The colors in the figure and equations help see the connection.

1.2 Alternating Direction Method of Multipliers

Given that all the conditions we defined for the f_i and f_α functions hold true, with the existence of prox_{g_α}, we can utilize alternating direction method of multipliers (ADMM) to tackle the inference problem in Eq. (1). The general idea of ADMM is based on augmented Lagriangian, in which we first rewrite the problem of minimizing Eq. (1) as

$$\arg\min_{\mathbf{y} \in \mathcal{Y}} \sum_i g_i(y_i, h_i(\mathbf{x}, \mathbf{w})) + \sum_\alpha h_\alpha(\mathbf{x}; \mathbf{w}) g_\alpha(z_\alpha) \;\; \text{s.t.} \;\; \forall \alpha, z_\alpha = \mathbf{w}_\alpha^T \mathbf{y}_\alpha$$

The augmented Lagrangian associated with the problem is then:

$$L(\mathbf{y}, \mathbf{z}, \mathbf{v}) = \sum_i g_i(y_i, h_i(\mathbf{x}, \mathbf{w})) + \sum_\alpha h_\alpha(\mathbf{x}; \mathbf{w}) g_\alpha(z_\alpha) + \mathbf{v}^T(\mathbf{W}^T \mathbf{y} - \mathbf{z}) + (\rho/2) \|\mathbf{W}^T \mathbf{y} - \mathbf{z}\|_2^2$$

where $\mathbf{W}$ is the matrix which is composed of column vectors of all $\mathbf{w}_\alpha$. Here, ρ is a hyper-parameter that controls the quadratic penalty and $\mathbf{v}$ is the dual variable associated with the consensus constraint (Lagrangian multiplier). We can then minimize the above energy function in an alternating manner:

$$\begin{cases} z_\alpha^{(n+1)} &= \text{prox}_{g_\alpha}(z_\alpha^{(n)} + \frac{\sigma_\rho}{h_\alpha(\mathbf{x}; \mathbf{w})}(\mathbf{w}_\alpha^T \mathbf{y}_\alpha^{(n)} + \frac{1}{\rho} v_\alpha^{(n)}) \\ y_i^{(n+1)} &= \text{prox}_{g_i, h_i(\mathbf{x}, \mathbf{w})}(y_i^{(n)} - \sigma_\tau(\mathbf{w}_{\cdot, i}^{*T}(\mathbf{z}^{(n+1)} - \mathbf{v}^{(n)})) \\ v_\alpha^{(n+1)} &= v_\alpha^{(n+1)} + \rho(\mathbf{w}_\alpha^T \mathbf{y}_\alpha^{(n+1)} - z_\alpha^{(n)}) \end{cases} \tag{3}$$

Again we can see that the iterative algorithm shares the same computation block with the proposed proximal structured network in Fig. 1. It is worth noting that although it seems that we have two convolution layers here, the convolution required for updating $z_\alpha^{(n+1)}$ and the one required for updating $v_\alpha^{(n)}$ are the same. Thus we still only need one convolution layer for our computation block, as the previous methods. Note that unlike the previous primal-dual method and proximal gradient method, we have a bias term in the two convolution layers in order to mimic ADMM.

1.3 Half-Quadratic Splitting

Given that all the conditions we defined for the f_i and f_α functions hold true, with g_i to be quadratic, we can utilize half-quadratic splitting (HQ) to tackle the inference problem in Eq. (1). The general idea of HQ is moving the high-order terms outside g_α expression, by gradually minimizing a series of the following new cost function:

$$\arg\min_{\mathbf{y}\in\mathcal{Y}}\min_{\mathbf{z}\in\mathcal{Z}}\sum_i g_i(y_i, h_i(\mathbf{x},\mathbf{w})) + \sum_\alpha h_\alpha(\mathbf{x};\mathbf{w})g_\alpha(z_\alpha) + \frac{\beta}{2}\sum_\alpha \|\mathbf{w}_\alpha^T\mathbf{y}_\alpha - z_\alpha\|_2^2$$

where $\mathbf{z}$ is the auxiliary variable. As the balancing parameter $\beta \to \infty$, the surrogate objective function converges to the original energy function in Eq. (1). The alternating optimization procedure thus contains steps including minimizing wrt $\mathbf{z}$ and wrt $\mathbf{y}$ respectively:

$$\begin{cases} z_\alpha^{(n+1)} &= \text{prox}_{g_\alpha}(z_\alpha^{(n)} + \frac{\beta\sigma_\rho}{h_\alpha(\mathbf{x};\mathbf{w})}(\mathbf{w}_\alpha^T\mathbf{y}_\alpha^{(n)}) \\ y_i^{(n+1)} &= \text{prox}_{g_i,h_i(\mathbf{x},\mathbf{w})}(y_i^{(n)} - \beta\sigma_\tau\mathbf{w}_{\cdot,i}^{*T}\mathbf{z}^{(n+1)}) \\ \bar{y}^{(n+1)} &= y^{(n+1)} + \sigma_{ex}(y^{(n+1)} - y^{(n)}) \end{cases} \tag{4}$$

where σ_u, σ_α and σ_{ex} are the hyperparameters represents unary, high-order and extrapolation gradient step respectively. If the extrapolation gradient step equals to 0, the above update rule corresponds to the standard half-quadratic splitting algorithm. Note that in this case with g_i to be quadratic, $\text{prox}_{g_i,h_i(\mathbf{x},\mathbf{w})}$ is a least-squares solver. From the colors in the figures and equations, we can see the connection between each step of the iteration and the building block of the proposed neural network in Fig. 1.

2 Additional Experimental Results

2.1 Depth Refinement

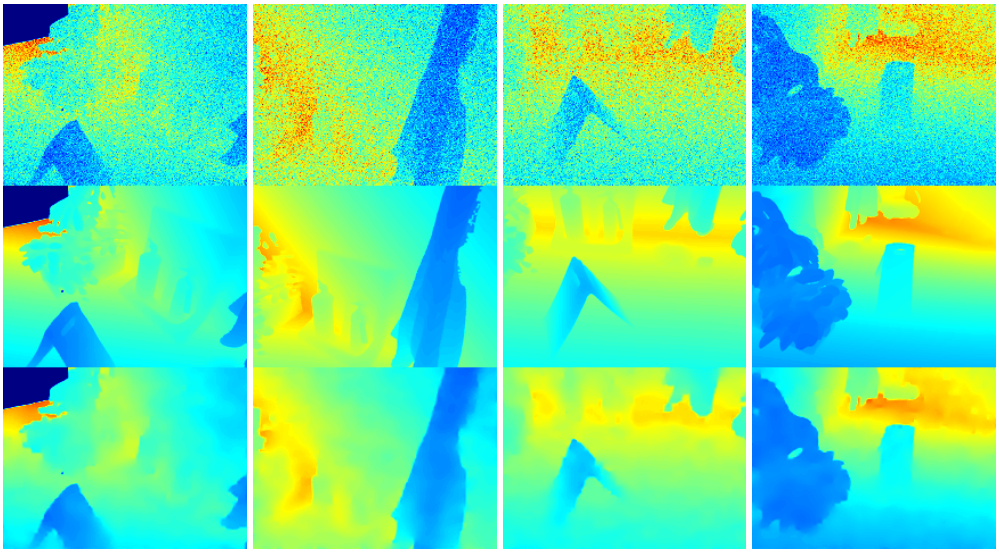

Figure 2: Additional depth refinement results. Top to bottom: noisy input, ground-truth depth, ours.

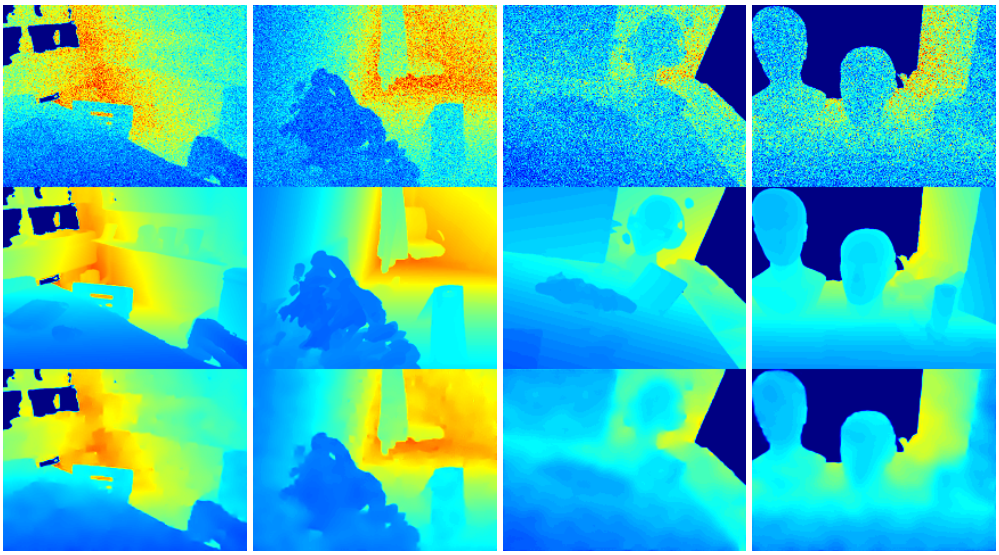

Figure 3: Additional depth refinement results. Top to bottom: noisy input, ground-truth depth, ours.

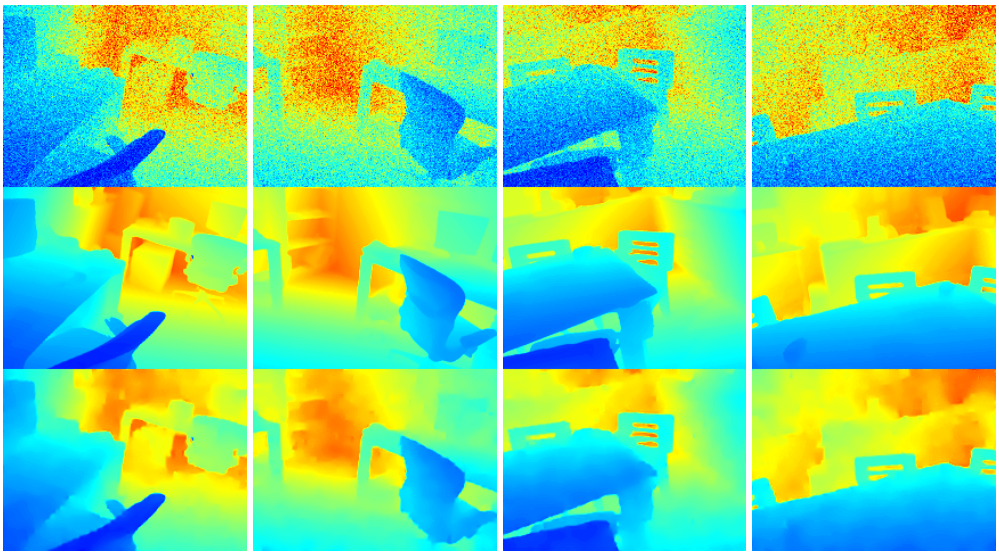

Figure 4: Additional depth refinement results. Top to bottom: noisy input, ground-truth depth, ours.

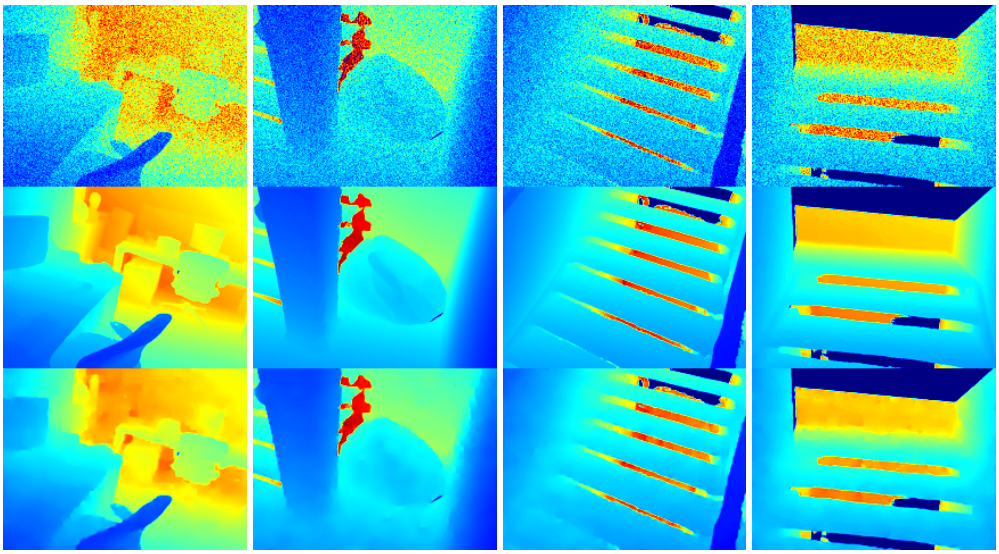

Figure 5: Additional depth refinement results. Top to bottom: noisy input, ground-truth depth, ours.

2.2 Natural Image Denoising

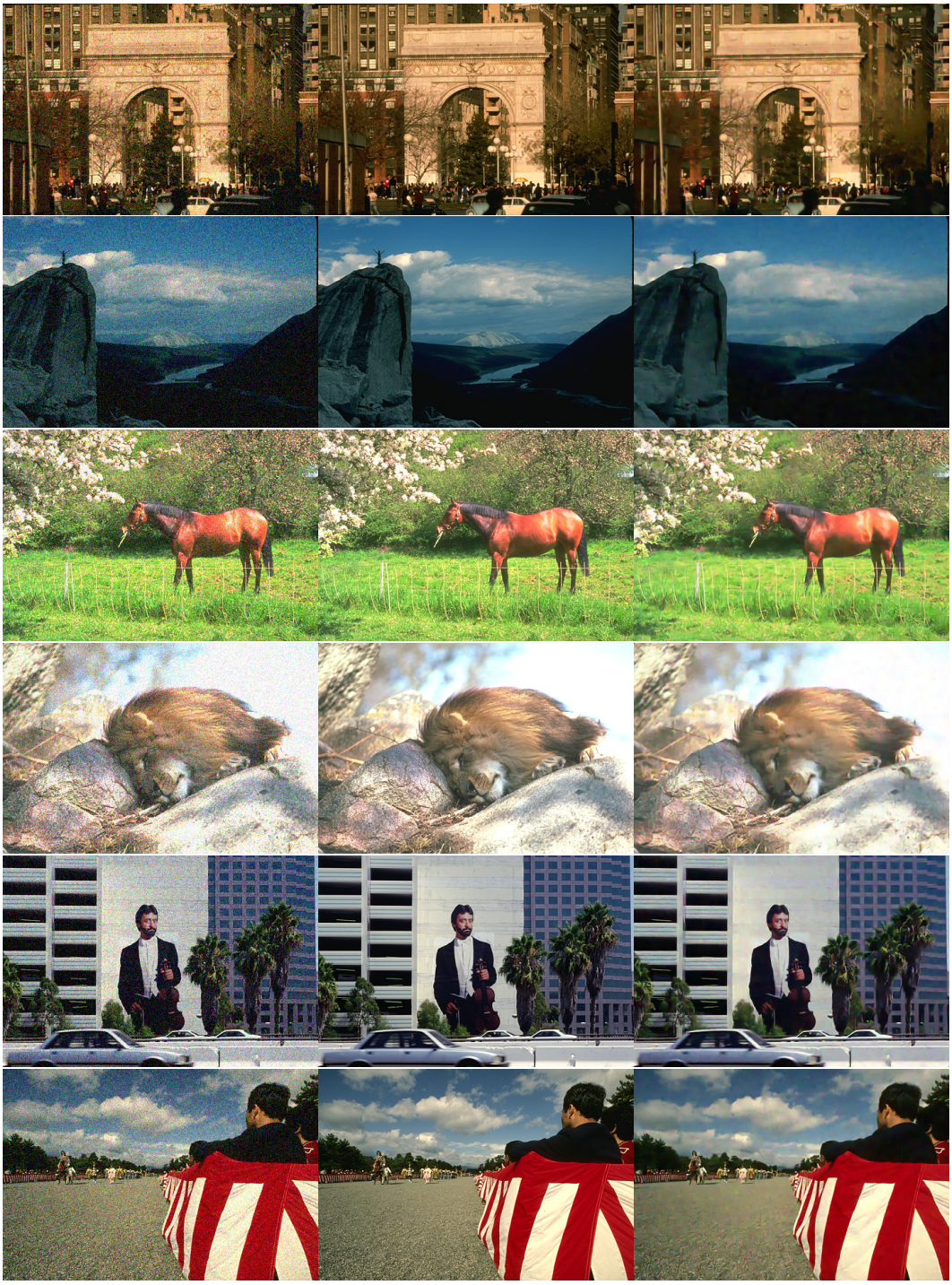

Figure 6: Additional natural image denoising results. Left to right: noisy input, ground-truth, ours.

2.3 Optical Flow

Figure 7: Additional optical flow results. Top to bottom: ground-truth optical flow, Flownet, ours.

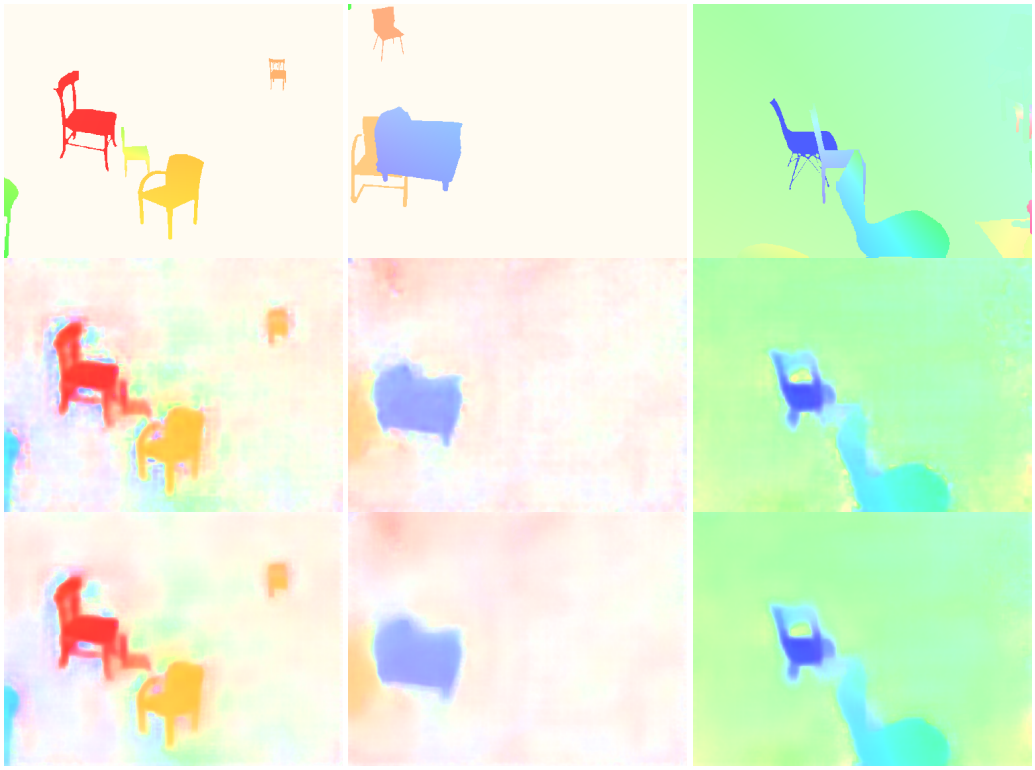

Figure 8: Additional optical flow results. Top to bottom: ground-truth optical flow, Flownet, ours.

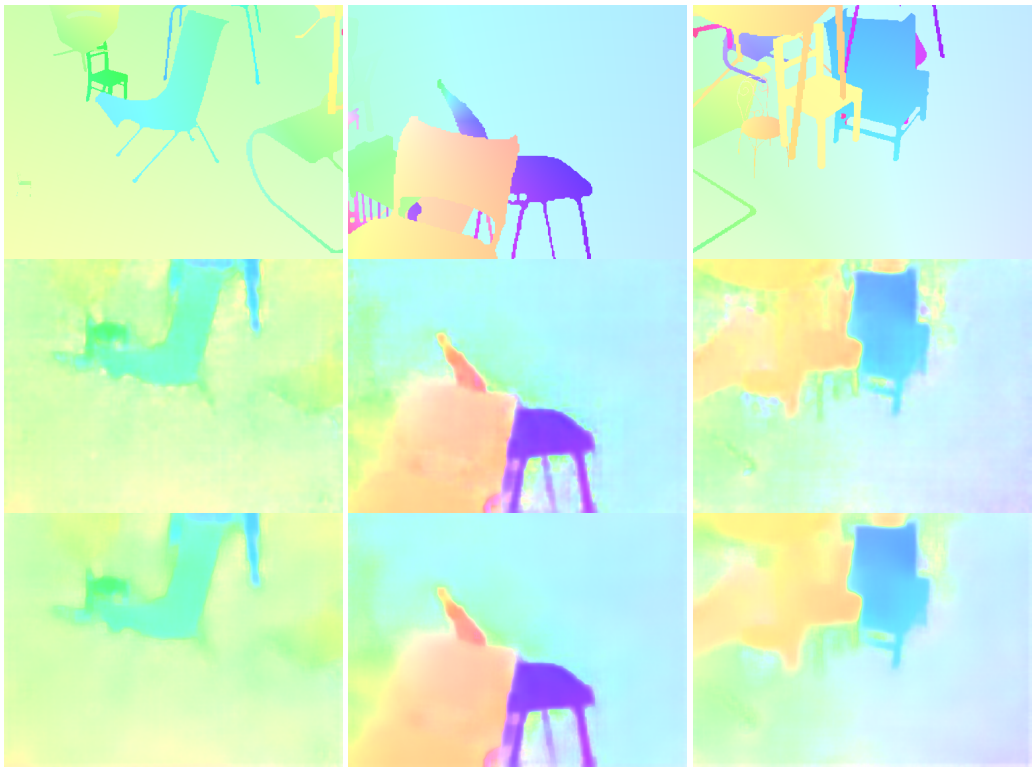

Figure 9: Additional optical flow results. Top to bottom: ground-truth optical flow, Flownet, ours.

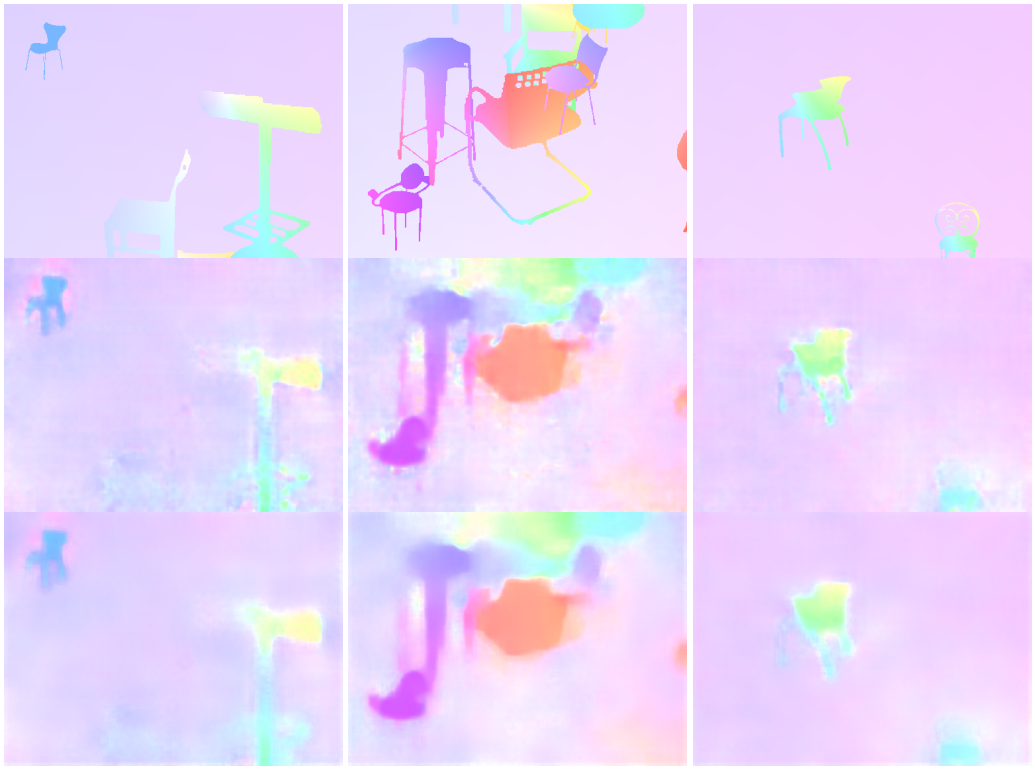

Figure 10: Additional optical flow results. Top to bottom: ground-truth optical flow, Flownet, ours.

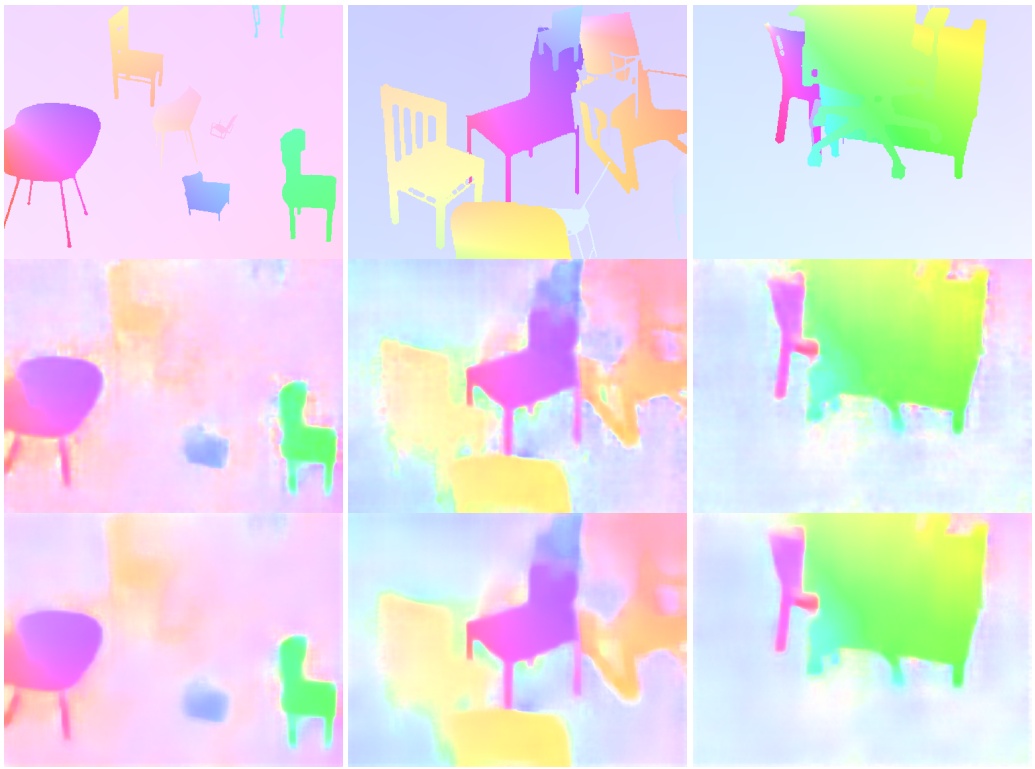

Figure 11: Additional optical flow results. Top to bottom: ground-truth optical flow, Flownet, ours.