Supplementary Material

A. Proofs

We first borrow the result from [11].

Proposition 1. *For the case of VP-SDE or DDPM sampling whose the forward diffusion is given by*

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}(t)}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}(t)}\boldsymbol{z}, \qquad \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{22}$$

$p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ has the unique posterior mean at

$$\hat{\boldsymbol{x}}_0 := \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] = \frac{1}{\sqrt{\bar{\alpha}(t)}}(\boldsymbol{x}_t + (1-\bar{\alpha}(t))\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)). \tag{23}$$

In our case, Proposition 1 holds for both the reverse conditional probability $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ as well as $p(\boldsymbol{k}_0|\boldsymbol{k}_t)$, as they are both constructed from DDPM. Given the posterior mean $\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{k}}_0$ that can be computed efficiently (i.e. via one forward pass through the neural network) during the intermediate steps, our proposal is to find a tractable approximation for $p(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t)$. Specifically, we propose the following approximation

$$p(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) \simeq p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{k}}_0), \quad \text{where} \quad \hat{\boldsymbol{x}}_0 := \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] = \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\boldsymbol{x}_0] \tag{24}$$

$$\hat{\boldsymbol{k}}_0 := \mathbb{E}[\boldsymbol{k}_0|\boldsymbol{k}_t] = \mathbb{E}_{\boldsymbol{k}_0 \sim p(\boldsymbol{k}_0|\boldsymbol{k}_t)}[\boldsymbol{k}_0] \tag{25}$$

Now, to quantify the approximation error induced by eq. (24),(25), the following definition is useful.

Definition 1 (Jensen gap [19, 50])**.** *Let $\boldsymbol{x}$ be a random variable with distribution $p(\boldsymbol{x})$. For some function f that may or may not be convex, the Jensen gap is defined as*

$$\mathcal{J}(f, \boldsymbol{x} \sim p(\boldsymbol{x})) = \mathbb{E}[f(\boldsymbol{x})] - f(\mathbb{E}[\boldsymbol{x}]), \tag{26}$$

where the expectation is taken over $p(\boldsymbol{x})$.

Using the Jensen gap defined in Definition 1, we attempt to achieve a meaningful upper bound on the gap. First, we have the following

Proposition 2 (Jensen gap upper bound [19])**.** *Define the absolute cenetered moment as $m_p := \sqrt[p]{\mathbb{E}[|X-\mu|^p]}$, and the mean as $\mu = \mathbb{E}[X]$. Assume that for $\alpha > 0$, there exists a positive number K such that for any $x \in \mathbb{R}$, $|f(x) - f(\mu)| \leq K|x-\mu|^\alpha$. Then,*

$$|\mathbb{E}[f(X) - f(\mathbb{E}[X])]| \leq \int |f(X) - f(\mu)| dp(X) \leq K \int |x-\mu|^\alpha dp(X) \leq M m_\alpha^\alpha. \tag{27}$$

The following lemmas from [11] are also useful.

Lemma 1. *Let $\phi(\cdot)$ be a univariate Gaussian density function with mean μ and variance σ^2. There exists a constant L such that $\forall x, y \in \mathbb{R}$,*

$$|\phi(x) - \phi(y)| \leq L|x-y|, \tag{28}$$

where $L = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2})$.

Lemma 2. *Let $\phi(\cdot)$ be an isotropic multivariate Gaussian density function with mean $\boldsymbol{\mu}$ and variance $\sigma^2\boldsymbol{I}$. There exists a constant L such that $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,*

$$\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \tag{29}$$

where $L = \frac{d}{\sqrt{2\pi\sigma^2}} e^{-1/2\sigma^2}$.

Theorem 1. *Under the same conditions in [11], we have*

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) \simeq \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t), \hat{\boldsymbol{k}}_0(\boldsymbol{k}_t))$$
$$\nabla_{\boldsymbol{k}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) \simeq \nabla_{\boldsymbol{k}_t} \log p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t), \hat{\boldsymbol{k}}_0(\boldsymbol{k}_t)).$$

Proof. The proof of the theorem is inspired by and builds upon [11]. We first note that $\boldsymbol{x}_t, \boldsymbol{k}_t \, \forall t \in [0, 1]$ are independent (See Fig. A.1). Further, $\boldsymbol{y}$ and $\boldsymbol{x}_t$ are conditionally independent on $\boldsymbol{x}_0$; $\boldsymbol{y}$ and $\boldsymbol{k}_t$ are conditionally independent on $\boldsymbol{k}_0$. Then, we have the following factorization

$$p(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) = \int p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k}_0) p(\boldsymbol{x}_0|\boldsymbol{x}_t) p(\boldsymbol{k}_0|\boldsymbol{k}_t) \, d\boldsymbol{x}_0 d\boldsymbol{k}_0 \tag{30}$$
$$= \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t), \, \boldsymbol{k}_0 \sim p(\boldsymbol{k}_0|\boldsymbol{k}_t)}[f(\boldsymbol{x}_0, \boldsymbol{k}_0)], \tag{31}$$

where $f(\boldsymbol{x}_0, \boldsymbol{k}_0) = h(\boldsymbol{k}_0 * \boldsymbol{x}_0)$, with $h(\boldsymbol{\mu})$ denoting the density function of an isotropic multivariate Gaussian density function with mean $\boldsymbol{\mu}$, and variance $\sigma^2 \boldsymbol{I}$. Our proposal is to use the Jensen approximation

$$p(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) \simeq p(\boldsymbol{y}|\mathbb{E}[\boldsymbol{x}_0, \boldsymbol{k}_0]) = p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{k}}_0), \tag{32}$$

where the last equality comes from the independency of $\boldsymbol{x}_0$ and $\boldsymbol{k}_0$. Now we derive the closed-form upper bound of the Jensen gap. For simplicity in exposition, let us define $\boldsymbol{K}_0 \boldsymbol{x}_0 \equiv \boldsymbol{k}_0 * \boldsymbol{x}_0, \equiv \boldsymbol{X}_0 \boldsymbol{k}_0$, where $\boldsymbol{K}_0, \boldsymbol{X}_0$ are block Hankel matrices that represent the convolution operation in matrix multiplication. Further, we denote $\|\bar{\boldsymbol{K}}_0\| := \mathbb{E}_{\boldsymbol{k}_0 \sim p(\boldsymbol{k}_0|\boldsymbol{k}_t)}[\|\boldsymbol{K}_0\|]$. Our Jensen gap reads

$$\mathcal{J}(f, p(\boldsymbol{x}_0|\boldsymbol{x}_t)p(\boldsymbol{k}_0|\boldsymbol{k}_t)) = |\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{k}_0}[f(\boldsymbol{x}_0, \boldsymbol{k}_0)] - f(\mathbb{E}_{\boldsymbol{x}_0}[\boldsymbol{x}_0], \mathbb{E}_{\boldsymbol{k}_0}[\boldsymbol{k}_0])| \tag{33}$$
$$\leq \underbrace{|\mathbb{E}_{\boldsymbol{k}_0, \boldsymbol{x}_0}[f(\boldsymbol{x}_0, \boldsymbol{k}_0)] - \mathbb{E}_{\boldsymbol{k}_0}[f(\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}_0], \boldsymbol{k}_0)]|}_{①} + \underbrace{|\mathbb{E}_{\boldsymbol{k}_0}[f(\mathbb{E}_{\boldsymbol{x}_0}[\boldsymbol{x}_0], \boldsymbol{k}_0)] - f(\mathbb{E}[\boldsymbol{x}_0], \mathbb{E}[\boldsymbol{k}_0])|}_{②}, \tag{34}$$

with

$$① = |\mathbb{E}_{\boldsymbol{k}_0}[\mathbb{E}_{\boldsymbol{x}_0}[f(\boldsymbol{x}_0, \boldsymbol{k}_0)] - f(\mathbb{E}_{\boldsymbol{x}_0}[\boldsymbol{x}_0], \boldsymbol{k}_0)]| \tag{35}$$
$$\overset{\text{(a)}}{\leq} \mathbb{E}_{\boldsymbol{k}_0}\left[\int |h(\boldsymbol{k}_0 * \boldsymbol{x}_0) - h(\boldsymbol{k}_0 * \hat{\boldsymbol{x}}_0)| dP(\boldsymbol{x}_0|\boldsymbol{x}_t)\right] \tag{36}$$
$$\overset{\text{(b)}}{\leq} \mathbb{E}_{\boldsymbol{k}_0}\left[\frac{d}{\sqrt{2\pi\sigma^2}} e^{-1/2\sigma^2} \int \|\boldsymbol{K}_0 \boldsymbol{x}_0 - \boldsymbol{K}_0 \hat{\boldsymbol{x}}_0\| dP(\boldsymbol{x}_0|\boldsymbol{x}_t)\right] \tag{37}$$
$$\leq \mathbb{E}_{\boldsymbol{k}_0}\left[\frac{d}{\sqrt{2\pi\sigma^2}} e^{-1/2\sigma^2} \|\boldsymbol{K}_0\| \int \|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0\| dP(\boldsymbol{x}_0|\boldsymbol{x}_t)\right] \tag{38}$$
$$\overset{\text{(c)}}{\leq} \mathbb{E}_{\boldsymbol{k}_0}\left[\frac{d}{\sqrt{2\pi\sigma^2}} e^{-1/2\sigma^2} \|\boldsymbol{K}_0\| m_{1, \boldsymbol{x}_0}\right] \tag{39}$$
$$\overset{\text{(d)}}{\leq} \frac{d}{\sqrt{2\pi\sigma^2}} e^{-1/2\sigma^2} \|\bar{\boldsymbol{K}}_0\| m_{1, \boldsymbol{x}_0}, \tag{40}$$

where (a) is from Proposition. 2, (b) is from Lemma. 2, and (c-d) are from the definitions. Moreover,

$$② \leq \int |h(\hat{\boldsymbol{k}}_0 * \hat{\boldsymbol{x}}_0) - h(\boldsymbol{k}_0 * \hat{\boldsymbol{x}}_0)| dP(\boldsymbol{k}_0|\boldsymbol{k}_t) \tag{41}$$
$$\leq \frac{d}{\sqrt{2\pi\sigma^2}} e^{-1/2\sigma^2} \int \|\hat{\boldsymbol{X}}_0 \boldsymbol{k}_0 - \hat{\boldsymbol{X}}_0 \hat{\boldsymbol{k}}_0\| dP(\boldsymbol{k}_0|\boldsymbol{k}_t) \tag{42}$$
$$\leq \frac{d}{\sqrt{2\pi\sigma^2}} e^{-1/2\sigma^2} \|\hat{\boldsymbol{X}}_0\| m_{1, \boldsymbol{k}_0}. \tag{43}$$

Hence

$$\mathcal{J}(f, p(\boldsymbol{x}_0|\boldsymbol{x}_t)p(\boldsymbol{k}_0|\boldsymbol{k}_t)) \leq \frac{d}{\sqrt{2\pi\sigma^2}} e^{-1/2\sigma^2} \left(\|\bar{\boldsymbol{K}}_0\| m_{1, \boldsymbol{x}_0} + \|\hat{\boldsymbol{X}}_0\| m_{1, \boldsymbol{k}_0}\right). \tag{44}$$

where

$$m_{1,\boldsymbol{x}_0} := \int \|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0\| dP(\boldsymbol{x}_0|\boldsymbol{x}_t) \tag{45}$$

$$m_{1,\boldsymbol{k}_0} := \int \|\boldsymbol{k}_0 - \hat{\boldsymbol{k}}_0\| dP(\boldsymbol{k}_0|\boldsymbol{k}_t) \tag{46}$$

We have derived that the approximation (32) has the Jensen gap upper bounded by (44). Finally, taking the derivative of the log to (32), we have that

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) \simeq \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t), \hat{\boldsymbol{k}}_0(\boldsymbol{k}_t))$$
$$\nabla_{\boldsymbol{k}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) \simeq \nabla_{\boldsymbol{k}_t} \log p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t), \hat{\boldsymbol{k}}_0(\boldsymbol{k}_t)).$$

Note that the approximation error from the Jensen gap approaches to zero as the noise level σ increase sufficiently. □

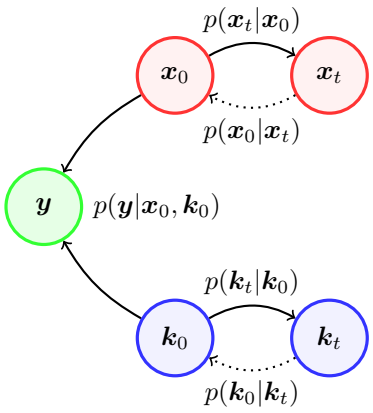


Figure A.1. Probabilistic graph of BlindDPS for blind deblurring.

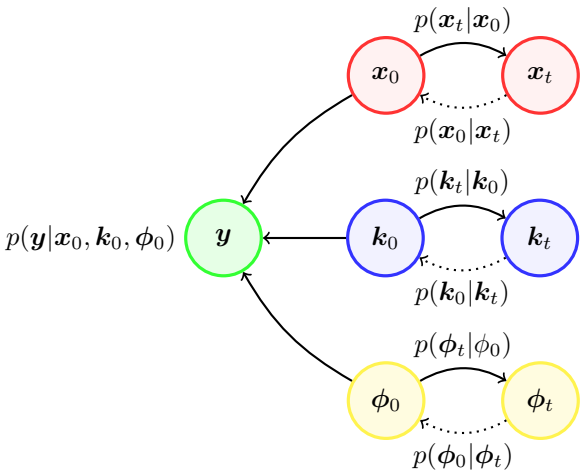


Figure B.1. Probabilistic graph of BlindDPS for imaging through turbulence.

B. BlindDPS

B.1. Imaging through turbulence

In terms of inverse problem solving, the tilt-blur model is often used [5, 6, 49], as the model is simple but fairly accurate. Specifically, we have

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k}_0, \boldsymbol{\phi}_0) := \mathcal{N}(\boldsymbol{y}|\boldsymbol{k}_0 * \mathcal{T}_{\boldsymbol{\phi}_0}(\boldsymbol{x}_0), \sigma^2 \boldsymbol{I}). \tag{47}$$

For details in the forward model that is used for our experiments, see Supplementary Section E. Note that the three factors are all independent, i.e.

$$p(\boldsymbol{x}_0, \boldsymbol{k}_0, \boldsymbol{\phi}_0|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k}_0, \boldsymbol{\phi}_0) p(\boldsymbol{x}_0) p(\boldsymbol{k}_0) p(\boldsymbol{\phi}_0).$$

Then, from Remark 1, we can again construct a system of reverse SDEs (See Fig. B.1) analogous to the blind deblurring case ((17),(18)):

$$\begin{aligned} d\boldsymbol{x} = (&-\frac{\beta(t)}{2}\boldsymbol{x} - \beta(t)[\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t), \hat{\boldsymbol{k}}_0(\boldsymbol{k}_t), \hat{\boldsymbol{\phi}}_0(\boldsymbol{\phi}_t)) \\ &+ \boldsymbol{s}^i_{\theta^*}(\boldsymbol{x}_t, t)])dt + \sqrt{\beta(t)}d\bar{\boldsymbol{w}}, \end{aligned} \tag{48}$$

$$\begin{aligned} d\boldsymbol{k} = (&-\frac{\beta(t)}{2}\boldsymbol{k} - \beta(t)[\nabla_{\boldsymbol{k}_t} \log p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t), \hat{\boldsymbol{k}}_0(\boldsymbol{k}_t), \hat{\boldsymbol{\phi}}_0(\boldsymbol{\phi}_t)) \\ &+ \boldsymbol{s}^k_{\theta^*}(\boldsymbol{k}_t, t)])dt + \sqrt{\beta(t)}d\bar{\boldsymbol{w}}, \end{aligned} \tag{49}$$

$$\begin{aligned} d\boldsymbol{\phi} = (&-\frac{\beta(t)}{2}\boldsymbol{k} - \beta(t)[\nabla_{\boldsymbol{\phi}_t} \log p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t), \hat{\boldsymbol{k}}_0(\boldsymbol{k}_t), \hat{\boldsymbol{\phi}}_0(\boldsymbol{\phi}_t)) \\ &+ \boldsymbol{s}^t_{\theta^*}(\boldsymbol{\phi}_t, t)])dt + \sqrt{\beta(t)}d\bar{\boldsymbol{w}}, \end{aligned} \tag{50}$$

where $\boldsymbol{s}^t_{\theta^*}$ is the score function trained to model the distribution of the tilt maps. Then, we can construct a similar method as shown in Algorithm 2 based on ancestral sampling analogous to Algorithm 1. Note that for solving imaging through turbulence, we do not use the ℓ_0 sparsity prior.

Algorithm 2 BlindDPS — Imaging through turbulence

Require: N, $\boldsymbol{y}$, α, $\{\tilde{\sigma}_i\}_{i=1}^N$
$\boldsymbol{x}_N, \boldsymbol{k}_N, \boldsymbol{\phi}_N \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
for $i = N-1$ **to** 0 **do**
  $\hat{\boldsymbol{s}}^i \leftarrow \boldsymbol{s}^i_{\theta^*}(\boldsymbol{x}_i, i)$
  $\hat{\boldsymbol{s}}^k \leftarrow \boldsymbol{s}^k_{\theta^*}(\boldsymbol{k}_i, i)$
  $\hat{\boldsymbol{s}}^k \leftarrow \boldsymbol{s}^t_{\theta^*}(\boldsymbol{\phi}_i, i)$
  $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{x}_i + \sqrt{1-\bar{\alpha}_i}\hat{\boldsymbol{s}}^i)$
  $\hat{\boldsymbol{k}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{k}_i + \sqrt{1-\bar{\alpha}_i}\hat{\boldsymbol{s}}^k)$
  $\hat{\boldsymbol{k}}_0 \leftarrow \mathcal{P}_C(\hat{\boldsymbol{k}}_0)$
  $\hat{\boldsymbol{\phi}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{\phi}_i + \sqrt{1-\bar{\alpha}_i}\hat{\boldsymbol{s}}^t)$
  $\boldsymbol{z}_i, \boldsymbol{z}_k, \boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
  $\boldsymbol{x}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{x}}_0 + \tilde{\sigma}_i \boldsymbol{z}_i$
  $\boldsymbol{k}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{k}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{k}}_0 + \tilde{\sigma}_i \boldsymbol{z}_k$
  $\boldsymbol{\phi}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{\phi}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{\phi}}_0 + \tilde{\sigma}_i \boldsymbol{z}_t$
  $\boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}'_{i-1} - \alpha\nabla_{\boldsymbol{x}_i}\|\boldsymbol{y} - \hat{\boldsymbol{k}}_0 * \mathcal{T}_{\boldsymbol{\phi}_0}(\hat{\boldsymbol{x}}_0)\|_2$
  $\boldsymbol{k}_{i-1} \leftarrow \boldsymbol{k}'_{i-1} - \alpha\nabla_{\boldsymbol{k}_i}\|\boldsymbol{y} - \hat{\boldsymbol{k}}_0 * \mathcal{T}_{\boldsymbol{\phi}_0}(\hat{\boldsymbol{x}}_0)\|_2$
  $\boldsymbol{\phi}_{i-1} \leftarrow \boldsymbol{\phi}'_{i-1} - \alpha\nabla_{\boldsymbol{\phi}_i}\|\boldsymbol{y} - \hat{\boldsymbol{k}}_0 * \mathcal{T}_{\boldsymbol{\phi}_0}(\hat{\boldsymbol{x}}_0)\|_2$
end for
return $\boldsymbol{x}_0, \boldsymbol{k}_0, \boldsymbol{\phi}_0$

C. Detailed Ablation Studies

C.1. Diffusion prior for the forward model

Let us revisit the Bayes' rule in the context of diffusion models for posterior sampling in blind deconvlution:

$$\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t, \boldsymbol{k}_t|\boldsymbol{y}) = \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) + \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$$
$$\nabla_{\boldsymbol{k}_t} \log p(\boldsymbol{x}_t, \boldsymbol{k}_t|\boldsymbol{y}) = \nabla_{\boldsymbol{k}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) + \nabla_{\boldsymbol{k}_t} \log p(\boldsymbol{k}_t).$$

We consider the case where we construct the diffusion prior for the image $\boldsymbol{x}$, but not for the kernel $\boldsymbol{k}$. In fact, this setting

Algorithm 3 Diffusion Posterior Sampling — Uniform prior

Require: $N, \boldsymbol{y}, \alpha_{\boldsymbol{x}}, \alpha_{\boldsymbol{k}}\{\tilde{\sigma}_i\}_{i=1}^{N}, \lambda, \sigma_{\text{init}}$

$\boldsymbol{x}_N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$\boldsymbol{k}_N \sim \text{GaussianKernel}(\sigma_{\text{init}})$
for $i = N-1$ **to** 0 **do**
  $\hat{\boldsymbol{s}}^i \leftarrow \boldsymbol{s}^i_{\theta^*}(\boldsymbol{x}_i, i)$
  $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{x}_i + \sqrt{1-\bar{\alpha}_i}\hat{\boldsymbol{s}}^i)$
  $\boldsymbol{k}_i \leftarrow \mathcal{P}_C(\boldsymbol{k}_i)$
  $\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\boldsymbol{x}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{x}}_0 + \tilde{\sigma}_i \boldsymbol{z}_i$
  $\boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}'_{i-1} - \alpha_{\boldsymbol{x}}\nabla_{\boldsymbol{x}_i}\|\boldsymbol{y} - \boldsymbol{k}_i * \hat{\boldsymbol{x}}_0\|_2$
  $\mathcal{L}_{\boldsymbol{k}} \leftarrow \|\boldsymbol{y} - \boldsymbol{k}_i * \hat{\boldsymbol{x}}_0\|_2 + \lambda \ell_0(\boldsymbol{k}_i)$
  $\boldsymbol{k}_{i-1} \leftarrow \boldsymbol{k}_i - \alpha_{\boldsymbol{k}}\nabla_{\boldsymbol{k}_i}\mathcal{L}_{\boldsymbol{k}}$
end for
return $\boldsymbol{x}_0, \boldsymbol{k}_0$

is similar to the concurrent work of Levac *et al.* [33], where the authors propose to use a score function only for the image, and not for the parameters for the motion artifact generating forward model. Note that the *blind* forward model setting here is considerably simpler than our method, since the parameter κ to be estimated is a scalar. In this regard, the authors propose to use a *uniform* prior for the unknown parameter κ, which makes the gradient of the prior to be simply 0, i.e. $\nabla_{\kappa_t} \log p(\kappa_t) = 0$. If we apply such uniform prior to our setting, our discretized update rule reads

$$\nabla_{\boldsymbol{x}_i} \log p(\boldsymbol{x}_i, \boldsymbol{k}_i|\boldsymbol{y}) \simeq \boldsymbol{s}^i_{\theta^*}(\boldsymbol{x}_i, i) - \frac{1}{\sigma^2}\nabla_{\boldsymbol{x}_i}\|\boldsymbol{y} - \boldsymbol{k}_i * \hat{\boldsymbol{x}}_0(\boldsymbol{x}_i)\|_2^2$$

$$\nabla_{\boldsymbol{k}_i} \log p(\boldsymbol{x}_i, \boldsymbol{k}_i|\boldsymbol{y}) \simeq -\frac{1}{\sigma^2}\nabla_{\boldsymbol{k}_i}\|\boldsymbol{y} - \boldsymbol{k}_i * \hat{\boldsymbol{x}}_0(\boldsymbol{x}_i)\|_2^2.$$

Additionally, similar to BlindDPS, one can further augment sparsity to the kernel estimation by using e.g. ℓ_0 regularization. Combined with the ancestral sampling steps, we arrive at Algorithm 3. Note that we chose Gaussian kernel as an initialization, but other choices are also feasible. The main difference between BlindDPS (Algorithm 1) and Algorithm 3 comes from the the complexity of the priors used. In order to quantify the performance gap, we chose 100 images from the FFHQ validation set, and compared the result of Algorithm 3 against BlindDPS. We performed grid search to find the optimal parameters $\alpha_{\boldsymbol{x}}, \alpha_{\boldsymbol{k}}, \lambda$, which were set to $\alpha_{\boldsymbol{x}} = 0.3$, $\alpha_{\boldsymbol{k}} = 0.3$, and $\lambda = 5.0$. In addition, to follow more closely to the method originally proposed in [33], we also include the case where the sparsity prior is set to zero by letting $\lambda = 0.0$, denoted as "Uniform prior" in Table. C.2.

Representative results can be seen in Fig. 7, and quantitative results can be found in Table C.2. Clearly, uniform prior far underperforms against the diffusion prior proposed in this work. We can conclude that while simple priors such as uniform prior may be a feasible option for *scalar* parameters, as the one in [33], much care should be taken when applied to higher dimensional parameters such as blind deconvolution.

	Motion				Gaussian			
λ	0.0	0.1	1.0	5.0	0.0	0.1	1.0	5.0
MNC ↑	0.929	0.956	0.958	0.959	0.996	0.997	0.996	0.997
MSE ↓	0.004	0.002	0.002	0.002	0.000	0.000	0.000	0.000
PSNR ↑	22.43	22.56	22.49	22.60	25.13	25.03	25.00	25.12
FID ↓	81.39	80.25	81.62	82.60	68.48	71.29	72.91	71.86
LPIPS ↓	0.281	0.279	0.281	0.277	0.228	0.230	0.232	0.231

Table C.1. Ablation study: effect of sparsity regularization in blind deconvolution.

	Image			Kernel	
Method	LPIPS ↓	PSNR ↑	SSIM ↑	MSE ↓	MNC ↑
BlindDPS (ours)	**0.247**	**23.65**	**0.786**	**0.002**	**0.958**
Uniform + Sparse	0.566	11.72	0.369	0.163	0.844
Uniform prior [33]	0.595	10.03	0.352	0.165	0.839

Table C.2. Ablation study: uniform prior vs. diffusion prior (BlindDPS).

C.2. Effect of sparsity regularization

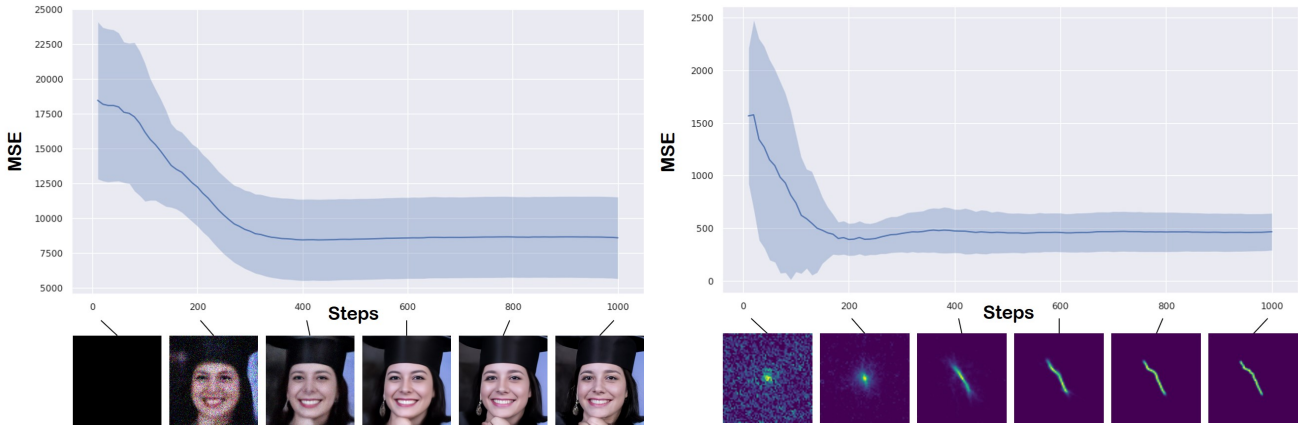
To check the effect of sparsity regularization in (20), we perform an ablation study by varying λ from 0.0 to 5.0. Specifically, we use l_1 sparsity regularization with different λ for 100 blurred images taken from validation set for FFHQ, with forward model and blur kernels adjusted to be identical to those of the main experiment (section E).

C.3. Progress of estimation

As discussed in section 3 of main text, the proposed method admits a natural Gaussian scale-space evolution of estimation, when visualized in the *denoised* representations $\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{k}}_0$. To quantify the trend in which the estimates evolve, we measure the MSE against the ground truth image and the kernel, and average the trend over 100 of the test data. We summarize the result in Fig. C.1a, C.1b. Here, we see that the MSE value drops to the minimum value at about 400/1000, 200/1000 iterations, which is relatively early in the whole reverse diffusion process. For the rest of the steps (especially for the images), the remaining high frequency details are in-filled, boosting the perceptual quality.

D. Extended Related Works

In this section, we discuss related works categorized into two applications that we tackle - blind deblurring, and imaging through turbulence.



(a) Progress of $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t)$ (b) Progress of $\hat{\boldsymbol{k}}_0(\boldsymbol{k}_t)$

Figure C.1. Progress of estimation error averaged over 100 test set in blind deconvolution. Blue line: mean value, shaded area: $\pm 1\sigma$. Measured with MSE against the ground truth.

D.1. Blind deblurring

We first review the optimization-based (model-based) methods that were extensively studied. The seminal work of Chan *et al.* [7] introduced the total variation (TV) prior, which enhances the gradient sparsity of both the image and the kernel. The scheme has been developed and re-invented over the years [34], yielding better practices to obtain stable results [45]. To promote sparsity of both the image and the kernel, regularizations based on ℓ_0 penalty [41], ℓ_p, $0 < p < 1$ penalty [64] based on the generalized iterative shrinkage algorithm (GISA) [63], ℓ_1, ℓ_2 [30] were proposed. Later on, it was shown that non-blurry natural images have sparse "dark channel" [43], where the dark channel is computed as the union of minimum values in patch occurrences. Promoting sparsity of the dark channel [44] has shown to be an effective method for performing blind deconvolution. When the regularization functions are chosen, one typically performs alternating optimization strategies [4] to solve the problem. It should be noted that it is often the case where the optimization strategy is non-trivial, and involves many tricks such as multi-scale optimization [42], and painful parameter tuning for specific input images. Wrong choice of parameter/optimization strategy typically results in heavily compromised performance.

In recent years, deep learning (DL) based methods have been largely developed. One can categorize DL methods into 1) explicit kernel estimation methods, where the network is designed to both deblur the image, and to estimate the exact kernel; 2) amortized inference, where the estimation of kernel does not take place. For the first type of methods, convolutional neural networks (CNN) were adopted for seperate modules, estimating the kernel and the deblurred image, respectively [48, 54, 59]. Advancing the conventional model-based priors, discriminative priors [35] and deep image priors (DIP) [47] were proposed, showing improved performance. While deep priors typically improves the performance, one should note that they are also often unstable, leading to undesirable solutions: both adversarial training and jointly training two deep image priors are hard to handle.

More recently, learning the inverse mapping without explicitly estimating the kernel has gained popularity. For these methods, neural network is trained through supervised learning with paired clean and blurry images. Especially, DeblurGAN [31] used the perceptual loss that helps to maintain contents and adversarial loss that minimizes the Wasserstein distance between the clean images and reconstructed images. DeblurGAN-v2 [32] focused on handling multi-scale features to solve the blind deblurring problem. They adopted Feature Pyramide Network (FPN) and proposed double-scale discriminators, where each discriminator measures the Wasserstein distance between clean images and reconstructed images at global and local patch level, respectively. Meanwhile, MPRNet [61] adopted a multi-stage learning method that decomposes the given problem into sub-problems and solves each one through a lightweight sub-network including a supervised attention module that gives weight to local features. As a result, blurry images are progressively restored. On the other hand, transformer based methods has been proposed and shown notable performance on deblurring task. Specifically, IPT [8] pretrained transformer on multiple image processing tasks and fine-tune the transformer on each tasks, Uformer [57] proposed LeWin transformer block for locally-enhanced self attention and multi-scale modulator, and Restormer [60] proposed two specialized transformer modules called MDTA and GDFN with progressive training scheme that enhances the image restoration performance on

different spatial resolutions. While often achieving state-of-the-art performance, these methods tend to compromise flexibility, modularity, and generalization capacity. For instance, the model cannot handle degradations that deviate from the traning data.

D.2. Imaging through turbulence

Although the correct estimation model for imaging through turbulence is tilt-then-blur [6], for inverse problem solving, the blur-then-tilt model is more often used. This is mainly due to the ease of applying off-the-shelf blind deblurring methods once the tilt is mitigated through, e.g. optical flow [37]. While in our work, we only consider single frame turbulence mitigation for simplicity, it is usually the case where we have multiple temporal frames that are degraded by random phase distortions. Hence, removing the tilt proceeds by e.g. temporal averaging [62], variational model [58], frame selection [1], etc. Moreover, when dealing with sequence of images, the "Lucky image fusion" step is often performed to find the reference image with the least amount of phase distortion. For details in such step, see, e.g. [18]. Once the distortion (tilt) is mitigated, the deblurring step is often performed with off-the-shelf algorithms [1,58,62]. However, as most off-the-shelf deblurring algorithms do not take into account the kernel priors specifically for turbulence, a more specified algorithm leveraging basis expansion [39] was proposed.

Similar to deblurring methods, various DL based methods have been proposed. Utilizing CNN to estimate the phase distortion map [38] was proposed. Moreover, supervised learning based on pairs of simulated atmospheric turbulence images have been proposed over the years. Transfer learning approach from pre-trained deblurring network was proposed [20]. Variants of generative adversarial network (GAN) based methods were also proposed [25, 46], leveraging the adversarial learning scheme to enhance the visual quality of the reconstructions. Recently, a method that uses physics-driven transformer architecture dubbed TurbNet [40] was proposed. To the best of our knowledge, none of the methods in the literature considered using unsupervised reconstruction scheme by utilizing the generative prior, as in our method. Although our method is developed upon a rather simplified forward model of imaging through turbulence, we believe our work establishes a proof of concept, and opens up a new are regarding turbulence reconstruction.

E. Inverse problem setting

In this section, we briefly summarize how our forward model is constructed.

E.1. Blind deblurring

The forward model is given as

$$\boldsymbol{y} = \boldsymbol{k}_0 * \boldsymbol{x}_0 + \boldsymbol{n},\ \boldsymbol{n} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}), \tag{51}$$

where $\sigma = 0.02$ is set as the measurement noise level. The size of the kernel is set to 64×64. For motion blur kernels, we use the random kernel generator from[1] with intensity value set to 0.5.

E.2. Imaging through turbulence

The forward model is given as

$$\boldsymbol{y} = \boldsymbol{k}_0 * \mathcal{T}_{\boldsymbol{\phi}_0}(\boldsymbol{x}_0) + \boldsymbol{n},\ \boldsymbol{n} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}), \tag{52}$$

where $\boldsymbol{\phi}$ is the tilt vector field that has identical size of the given image (i.e. in our case 256×256). Specifically, the tilt vector field is generated with the algorithm proposed in [5]. The parameters are set to $M = 500, N = 32, \sigma = 1.0$, with all the other parameters set same to the baseline. The blur kernel $\boldsymbol{k}_0$ is taken to be isotropic Gaussian kernel with standard deviation of 0.4 (FFHQ), and 0.2 (ImageNet). The proposed algorithm for solving imaging through turbulence is presented in Algorithm. 2.

F. Experimental Details

F.1. Blind deblurring

For blind deblurring, we conduct experiments on FFHQ 256×256 [26], and AFHQ-dog 256×256 [10] dataset on {motion, Gaussian}-deblurring. We choose 1k validation set for FFHQ, and use 500 test sets for AFHQ-dog. We leverage pre-trained score functions, as in the experimental setting of [13]. We train the score function on 60k generated blur kernels of size 64×64 (both Gaussian and motion[2]) for 3M steps with a small U-Net [16]. For testing, motion blur kernel is randomly generated with intensity 0.5 following [11], and the standard deviation of the gaussian kernels is set to 3.0. Step size for Algorithm 1 is set to $\alpha = 0.3$ for both FFHQ/AFHQ. We choose $R_{\boldsymbol{k}}(\cdot) = \ell_1, \lambda = 1.0$ for FFHQ, and $R_{\boldsymbol{k}}(\cdot) = \ell_0, \lambda = 5.0$ for AFHQ.

F.2. Imaging through turbulence

For imaging through turbulence, we conduct experiments with FFHQ 256×256, and ImageNet 256×256 [15], with pre-trained ImageNet score function taken from [16]. The score function for kernel blur is taken from the blind deblurring experiment, and the score function for the tilt map is trained with 50k randomly generated tilt maps following [5]. The point spread function (PSF) is assumed to

[1] https://github.com/LeviBorodenko/motionblur/blob/master/motionblur.py

[2] https://github.com/topics/motion-blur

be a Gaussian with standard deviation of 4.0, 2.0 for FFHQ, ImageNet, respectively (size 64×64). For both blind inverse problems, we add Gaussian measurement noise with $\sigma = 0.02$. Step size is set to $\alpha = 0.3$.

F.3. Training

We take pre-trained score function for the FFHQ dataset, and the ImageNet dataset, following the settings of [11]. When training the score function for kernels, we create a database of that consists of 60k 64×64 kernels. Among them, 50k motion blur kernels were generated from[3], by sampling the intensity value $I \sim \text{Unif}(0.2, 1.0)$. The other 10k Gaussian blur kernels were generated with random standard deviation $\sigma \sim \text{Unif}(0.1, 5.0)$.

For training the score function for kernel / tilt-map, we use the U-Net architecture from `guided-diffusion`[4], and train the models using base configurations. The models were trained with a single RTX 3090 GPU for 3.0M / 1.5M steps, which took about one day / two days, respectively.

F.4. Compute time

As stated in the limitations, the number of score functions that are used at inference time scales linearly with the number of components involved in the forward model. For blind deblurring, two neural networks are used (image, kernel), and for imaging through turbulence, three neural networks are used (image, kernel, tilt map). In order to quantify additional compute cost in each of the situation, we measure the wall-clock time to reconstruct a single image with a single RTX 2080ti GPU. DPS [11]: 132.39 sec. BlindDPS—Blind deblurring(2 score functions): 180.22 sec. BlindDPS—Imaging through turbulence(3 score functions): 220.76 sec. For all the iterative comparison methods that follow, we also measure the wall-clock time for compute measured with a single RTX 2080ti GPU.

F.5. Comparison methods

For blind deblurring, we compare the reconstruction performance of BlindDPS against state-of-the-art methods. Specifically, we choose MPRNet [61] and DeblurGANv2 [32] as supervised learning-based baselines that are incapable of kernel estimation, but work through amortized inference. We also compare our method against SelfDeblur [47], which leverages deep image prior (DIP) for estimating both the kernel and the image. For optimization-based methods, we use Pan-dark channel prior (Pan-DCP) [44], Pan-ℓ_0 [41], and Perrone *et al* [45]. For imaging through turbulence, we use MPRNet [61], DeblurGANv2 [32], and TSR-WGAN [25] as comparison methods that are based on supervised traning. We also compare against ILVR [9], which is a diffusion model-based method that is capable of restoring images from low resolution.

Pan-DCP [44]. The method utilizes the dark channel prior as the regularization function for images. We use the official implementation[5], with the parameters advised for facial blur images. We list the specific parameters below. Optimization is performed in a coarse-to-fine strategy in 8 different stages. Compute time: 132.19 sec.

- $\lambda_{\text{dark}} = 4e-3$
- $\lambda_{\text{grad}} = 4e-3$
- $\lambda_{\text{tv}} = 1e-3$
- $\lambda_{\text{l0}} = 5e-4$

Pan-ℓ_0 [41]. The method regularizes ℓ_0 regularization for both the image and the kernel. We use the official implementation[6], with the parameters set as below. Optimization and post-processing is performed similar to Pan-DCP. Compute time: 151.01 sec.

- $\lambda_{\text{pixel}} = 4e-3$
- $\lambda_{\text{grad}} = 4e-3$
- $\lambda_{\text{tv}} = 1e-3$
- $\lambda_{\text{l0}} = 2e-3$

SelfDeblur [47]. We use the default setting of YCbCr deblurring that selfdeblur uses, with static learning rate of 0.01 for 2500 steps. Optimization is performed by minimizing the MSE for the first 500 steps, and then switching the loss to $1 - SSIM(\cdot, \cdot)$. Compute time: 142.91 sec.

MPRNet [61]. We use the official implementation[7], with the parameters, learning rate decay and neural network architectures advised for the deblurring task. For both FFHQ and AFHQ, we train the model for 30k iterations with a batch size of 3. For a fair comparison with the proposed method, half of the input image consists of gaussian blurred images and the other half image consists of motion blurred image.

DeblurGANv2 [32]. We use the official implementation[8], by following the default settings for parameters, data augmentation strategies and neural network architectures. Specifically, we train the model by minimizing sum of pixel distance loss, WGAN-gp adversarial loss and perceptual loss with weight parameters as below. Inception-ResNet-v2 is used as backbone of the generator. For both FFHQ and

[3] https://github.com/LeviBorodenko/motionblur

[4] https://github.com/openai/guided-diffusion

[5] https://jspan.github.io/projects/dark-channel-deblur/index.html

[6] https://jspan.github.io/projects/text-deblurring/index.html

[7] https://github.com/swz30/MPRNet

[8] https://github.com/VITA-Group/DeblurGANv2

AFHQ, we train the model for 1.5 million iterations with a batch size of 1 and input image contains half Gaussian blurred images and the other half motion blurred images for fair comparison with the proposed method.

- $\lambda_{\text{pixel}} = 5e-1$
- $\lambda_{\text{adv}} = 6e-3$
- $\lambda_{\text{perceptual}} = 1e-2$

ILVR [9]. We choose the following hyper-paremters: down-scaling factor of 16, 1000 sampling steps, with the latent guidance applied for 1000-100 sampling steps. We use the same score functions that were used for BlindDPS.

TSR-WGAN [25]. The original work considers spatio-temporal 3D data, whereas our inverse problem setting considers single frame imaging through turbulence. Hence, we design a U-Net like network architecture that consists of 2D convolutions rather than leveraging 3D convolutions. Other training configurations follow the default setting of [25].

Note that for methods that are capable of estimating the kernel simultaneously (i.e. Pan-DCP, Pan-ℓ_0, SelfDeblur), only odd-sized kernels can be estimated, whereas our ground truth kernels are even-sized. To match the discrepancy, we estimate 65×65 sized kernel first, and then cut the redundant row/column as the post-processing step. In practice, such discrepancy only affects the result marginally.

G. Further Experiments

Further experimental results on blind deblurring are shown in Fig. G.1, G.2, G.3, G.4. Further experimental results on imaging through turbulence are shown in Fig. G.5, G.6.

Figure G.1. Blind motion deblurring results on the FFHQ 256 $\times$ 256 dataset. (a) Measurement, (b) Pan-DCP [44], (c) MPRNet [61], (d) SelfDeblur [47], (e) BlindDPS (**ours**), (f) Ground truth.

Figure G.2. Blind motion deblurring results on the AFHQ 256 $\times$ 256 dataset. (a) Measurement, (b) Pan-DCP [44], (c) MPRNet [61], (d) SelfDeblur [47], (e) BlindDPS (**ours**), (f) Ground truth.

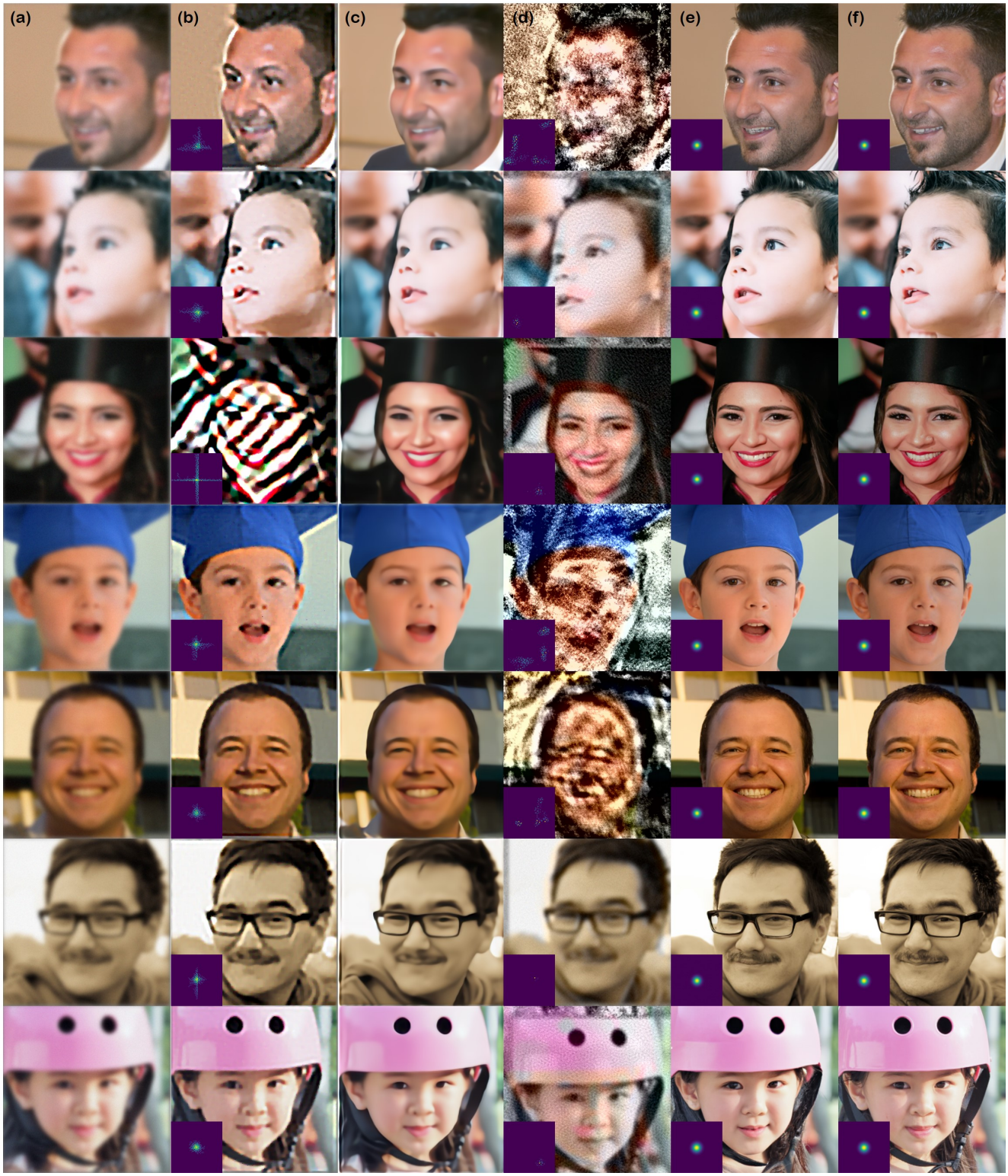


Figure G.3. Blind Gaussian deblurring results on the FFHQ 256 × 256 dataset. (a) Measurement, (b) Pan-DCP [44], (c) MPRNet [61], (d) SelfDeblur [47], (e) BlindDPS (**ours**), (f) Ground truth.

Figure G.4. Blind Gaussian deblurring results on the AFHQ 256×256 dataset. (a) Measurement, (b) Pan-DCP [44], (c) MPRNet [61], (d) SelfDeblur [47], (e) BlindDPS (**ours**), (f) Ground truth.

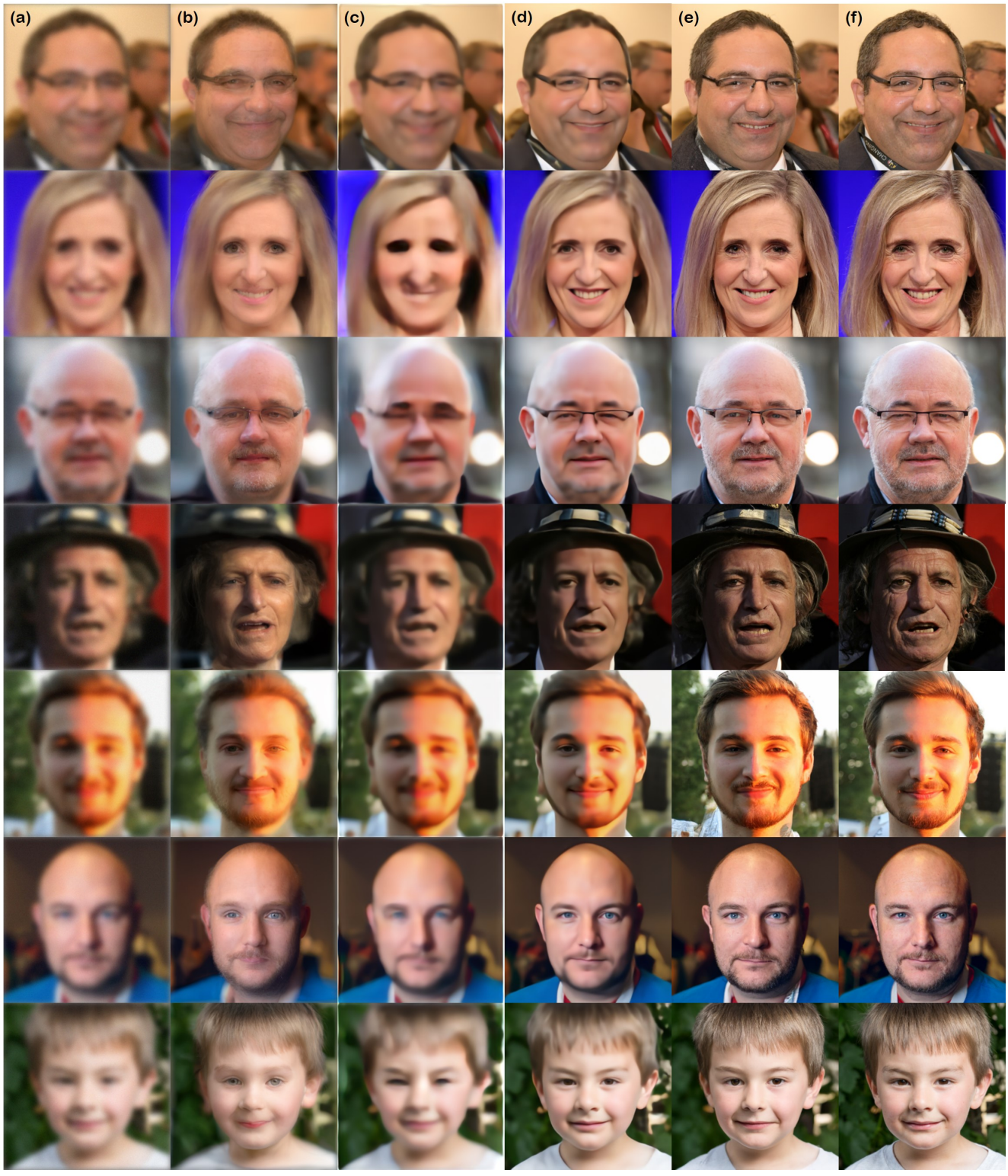


Figure G.5. Imaging through turbulence results on the FFHQ 256 $\times$ 256 dataset. (a) Measurement, (b) ILVR [9], (c) MPRNet [61], (d) TSR-WGAN [25], (e) BlindDPS (**ours**), (f) Ground truth.

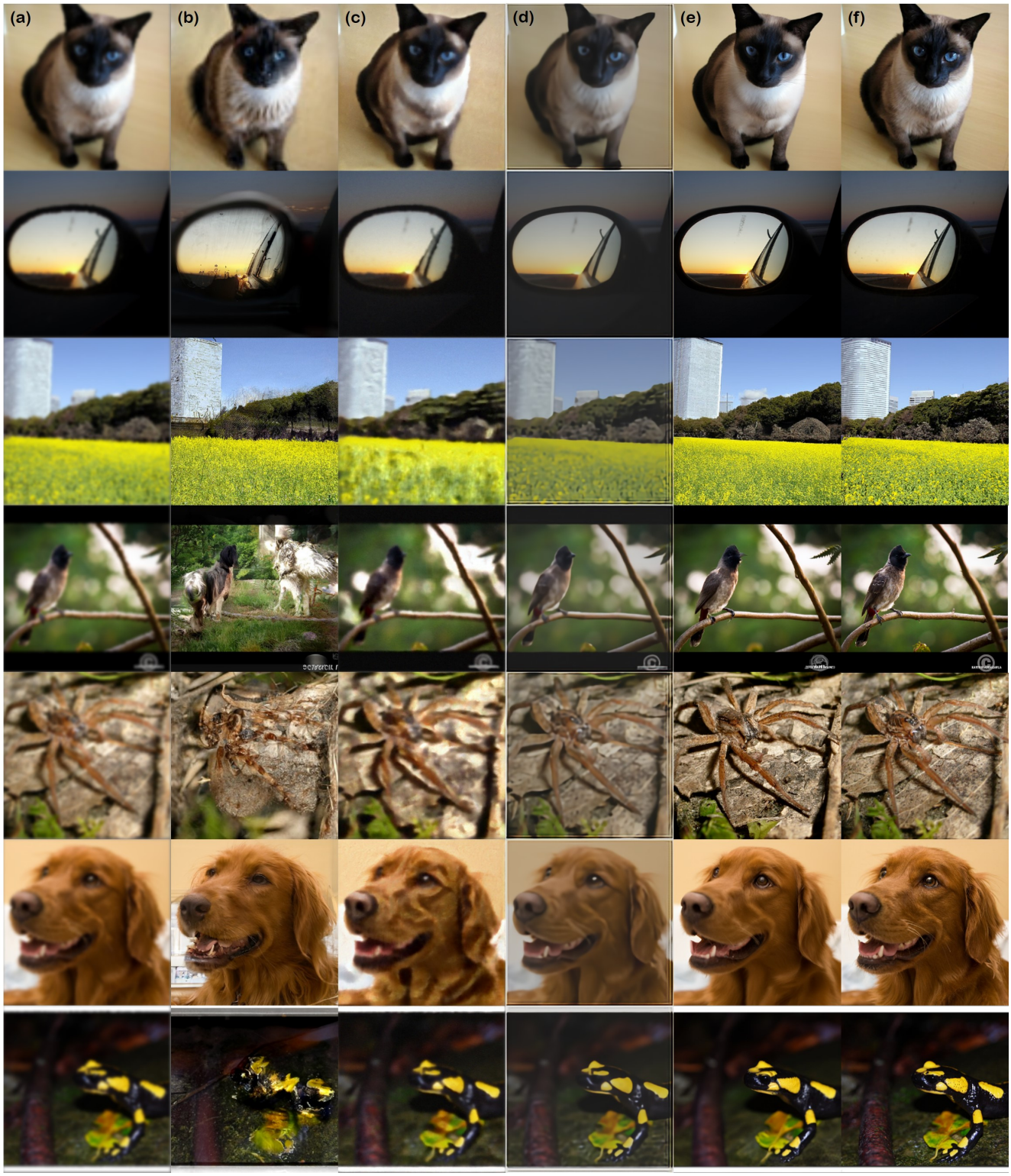


Figure G.6. Imaging through turbulence results on the ImageNet 256 × 256 dataset. (a) Measurement, (b) ILVR [9], (c) MPRNet [61], (d) TSR-WGAN [25], (e) BlindDPS (**ours**), (f) Ground truth.