Explore Image Deblurring via Blur Kernel Space

Phong Tran¹
Anh Tran¹, ²
Quynh Phung¹
Minh Hoai¹, ³

¹VinAI Research, Hanoi, Vietnam, ²VinUniversity, Hanoi, Vietnam, ³Stony Brook University, Stony Brook, NY 11790, USA

https://github.com/VinAIResearch/blurr-kernel-space-exploring
Image Deblurring

Moving object

Camera shaking
Image Deblurring

Deblurring method
MAP-based Methods

\[ y = x \ast k + n \]

- \( y \): blur image
- \( x \): sharp image
- \( k \): blur kernel
- \( n \): noise

Linear and uniform
MAP-based Methods

MAP Framework:

\[ x, k = \arg\max_{x,k} P(y|x, k)P(x)P(k) \]
MAP-based Methods

Gradient-based penalty, dark channels, ...

Linear and uniform

*  

Sparsity, Spectral properties, ...
MAP-based Methods

- Gradient-based penalty, dark channels, ...
- Sparsity, Spectral properties, ...
- Linear and uniform kernel

Does not hold in general
Deep Learning Models
Deep Learning Models - Challenges

Kernel overfitting

CNN
Our Work

- Generalize MAP-based method
- Leverage neural networks
Our Work

Assumptions: \[ y = \mathcal{F}(x, k) \]

\[ \mathcal{F}(\cdot, k) : \text{Blur operator parameterized by } k \]
Our Work

Assumptions:

\[ y = \mathcal{F}(x, k) \]

\( \mathcal{F}(\cdot, k) \) : Blur operator parameterized by k

\( \mathcal{G}(x, y) \) : Extract blur kernel k from (x, y)
Our Work

Find F and G
Our Work

Find F and G

Blind Deblurring
Our Work

Find F and G
Blind Deblurring
Blur Synthesis
Kernel Encoding

- F and G are implemented by two neural networks.

- For \((x, y) \sim P_{\text{data}}(x, y)\), F and G are jointly optimized by minimizing the objective function:
  \[
  \mathbb{E}_{x,y} \left[ \rho(y, \mathcal{F}(x, \mathcal{G}(x, y))) \right]
  \]
Kernel Encoding

• F and G are implemented by two neural networks.

• For \((x, y) \sim P_{\text{data}}(x, y)\). F and G are jointly optimized by minimizing the objective function:

\[
\mathbb{E}_{x,y} [\rho(y, \mathcal{F}(x, \mathcal{G}(x, y)))]
\]

Charbonnier Loss

Recon blurry image
Generic Image Deblurring

- $X$ and $k$ are alternatively optimized by minimizing:

$$\sum_{i=1}^{n} \rho(y_i, F(x_i, G(x_i, y_i)))$$

Charbonnier Loss

Recon blurry image
• X and k are alternatively optimized by minimizing:

\[ \sum_{i=1}^{n} \rho(y_i, F(x_i, G(x_i, y_i))) \]

---

**Algorithm 1** Blind image deblurring

**Input:** blurry image \( y \)

**Output:** sharp image \( x \)

1. Sample \( z_x \sim \mathcal{N}(0, I) \)
2. Randomly initialize \( \theta_x \) of \( G_{\theta_x} \)
3. **while** \( \theta_x \) has not converged **do**
   4. Sample \( z_k \sim \mathcal{N}(0, I) \)
   5. Randomly initialize \( \theta_k \) of \( G_{\theta_k} \)
   6. **while** \( \theta_k \) has not converged **do**
      7. \( g_k \leftarrow \partial \mathcal{L}(\theta_x, \theta_k)/\partial \theta_k \)
      8. \( \theta_k \leftarrow \theta_k + \alpha \ast ADAM(\theta_k, g_k) \)
   9. **end while**
   10. \( g_x \leftarrow \partial \mathcal{L}(\theta_x, \theta_k)/\partial \theta_x \)
   11. \( \theta_x \leftarrow \theta_x + \alpha \ast ADAM(\theta_x, g_x) \)
12. **end while**
13. \( x = G_{\theta_x}(z_x) \)
Generic Image Deblurring

X and k are alternatively optimized by minimizing:

$$\sum_{i=1}^{n} \rho(y_i, F(x_i, G(x_i, y_i)))$$

Algorithm 1 Blind image deblurring

**Input:** blurry image y

**Output:** sharp image x

1: Sample $z_x \sim \mathcal{N}(0, I)$
2: Randomly initialize $\theta_x$ of $G_{\theta_x}^x$
3: while $\theta_x$ has not converged do
4:    Sample $z_k \sim \mathcal{N}(0, I)$
5:    Randomly initialize $\theta_k$ of $G_{\theta_k}^k$
6:    while $\theta_k$ has not converged do
7:       $g_k \leftarrow \partial \mathcal{L}(\theta_x, \theta_k)/\partial \theta_k$
8:       $\theta_k \leftarrow \theta_k + \alpha \ast ADAM(\theta_k, g_k)$
9:    end while
10:   $g_x \leftarrow \partial \mathcal{L}(\theta_x, \theta_k)/\partial \theta_x$
11:  $\theta_x \leftarrow \theta_x + \alpha \ast ADAM(\theta_x, g_x)$
12: end while
13: $x = G_{\theta_x}(z_x)$
Generic Image Deblurring

- $X$ and $k$ are alternatively optimized by minimizing:

$$\sum_{i=1}^{n} \rho(y_i, F(x_i, G(x_i, y_i)))$$

```
Algorithm 1 Blind image deblurring

Input: blurry image $y$
Output: sharp image $x$

1: Sample $z_x \sim \mathcal{N}(0, I)$
2: Randomly initialize $\theta_x$ of $G_{\theta_x}^x$
3: while $\theta_x$ has not converged do
4: Sample $z_k \sim \mathcal{N}(0, I)$
5: Randomly initialize $\theta_k$ of $G_{\theta_k}^k$
6: while $\theta_k$ has not converged do
7: $g_k \leftarrow \partial L(\theta_x, \theta_k)/\partial \theta_k$
8: $\theta_k \leftarrow \theta_k + \alpha \ast ADAM(\theta_k, g_k)$
9: end while
10: $g_x \leftarrow \partial L(\theta_x, \theta_k)/\partial \theta_x$
11: $\theta_x \leftarrow \theta_x + \alpha \ast ADAM(\theta_x, g_x)$
12: end while
13: $x = G_{\theta_x}(z_x)$
```
Generic Image Deblurring

- X and k are alternatively optimized by minimizing:

\[
\rho(y, F(x, k)) + \lambda \|k\|_2 + \gamma (g_u^2(x) + g_v^2(x))^{\alpha/2}
\]

**Algorithm 1** Blind image deblurring

**Input:** blurry image y  
**Output:** sharp image x

1: Sample \( z_x \sim \mathcal{N}(0, I) \)
2: Randomly initialize \( \theta_x \) of \( G_{\theta_x}^x \)
3: **while** \( \theta_x \) has not converged **do**
4: \hspace{1em} Sample \( z_k \sim \mathcal{N}(0, I) \)
5: \hspace{1em} Randomly initialize \( \theta_k \) of \( G_{\theta_k}^k \)
6: \hspace{1em} **while** \( \theta_k \) has not converged **do**
7: \hspace{2em} \( g_k \leftarrow \partial L(\theta_x, \theta_k) / \partial \theta_k \)
8: \hspace{2em} \( \theta_k \leftarrow \theta_k + \alpha \ast \text{ADAM}(\theta_k, g_k) \)
9: \hspace{1em} **end while**
10: \hspace{1em} \( g_x \leftarrow \partial L(\theta_x, \theta_k) / \partial \theta_x \)
11: \hspace{1em} \( \theta_x \leftarrow \theta_x + \alpha \ast \text{ADAM}(\theta_x, g_x) \)
12: **end while**
13: \( x = G_{\theta_x}(z_x) \)
**Generic Image Deblurring**

- Deep Image Prior:
  - Replace $x$ by $G_{\theta_x}^x$
  - Replace $k$ by $G_{\theta_k}^k$

- $x$ and $k$ are alternatively optimized by minimizing:
  \[
  \rho(y, F(x, k)) + \lambda \|k\|_2 + \gamma(g_u^2(x) + g_v^2(x))^{\alpha/2}
  \]

---

**Algorithm 1** Blind image deblurring

**Input:** blurry image $y$

**Output:** sharp image $x$

1. Sample $z_x \sim \mathcal{N}(0, I)$
2. Randomly initialize $\theta_x$ of $G_{\theta_x}^x$
3. While $\theta_x$ has not converged do
   4. Sample $z_k \sim \mathcal{N}(0, I)$
   5. Randomly initialize $\theta_k$ of $G_{\theta_k}^k$
   6. While $\theta_k$ has not converged do
      7. $g_k \leftarrow \partial\mathcal{L}(\theta_x, \theta_k)/\partial\theta_k$
      8. $\theta_k \leftarrow \theta_k + \alpha \ast ADAM(\theta_k, g_k)$
      9. End while
   10. $g_x \leftarrow \partial\mathcal{L}(\theta_x, \theta_k)/\partial\theta_x$
   11. $\theta_x \leftarrow \theta_x + \alpha \ast ADAM(\theta_x, g_x)$
   12. End while
13. $x = G_{\theta_x}(z_x)$
Domain-specific Image Deblurring

\[ z^*, k^* = \arg \max_{z,k} \rho \left( F(G_{\text{style}}(z), k), y \right) + R_z(z) + R_k(k) \]

Regularization term
Blur Synthesis

\[ G(x_1, y_1) \]

\((x_1, y_1)\)
Blur Synthesis

\[ G(x_1, y_1) \]

\[ F(x_2, k_1) \]

\( (x_1, y_1) \)

\( x_2 \)

\( y_2 \)
Experimental Results – Kernel Encoding

Kernel 8

\((x_1, y_1)\)

\((x_2, y_2)\)
Experimental Results – Kernel Encoding

\[ G(x_1, y_1) \]

\[ F(x_2, k_1) \]

\[ x_2 \]

\[ y'_2 \]
Experimental Results – Kernel Encoding

\[ G(x_1, y_1) \]

\[ F(x_2, k_1) \]

\[ x_2 \]

\[ y'_2 \]

\[ y_2 \]

PSNR
### Experimental Results – Kernel Encoding

| PSNR (db)   | kernel 1 | kernel 2 | kernel 3 | kernel 4 |
|-------------|----------|----------|----------|----------|
|             | 49.48    | 51.93    | 52.06    | 53.74    |
| PSNR (db)   | kernel 5 | kernel 6 | kernel 7 | kernel 8 |
|             | 49.91    | 49.49    | 51.43    | 50.38    |

Blur transferring performance on Levin dataset
### Experimental Results – Kernel Encoding

| Training data   | Dataset   |        |        |
|-----------------|-----------|--------|--------|
|                 | REDS      | GOPRO  |        |
| Original        | 30.70     | 30.20  |        |
| Blur-swapped    | 29.43     | 28.49  |        |

SRN performance when training on blur-swapped dataset
Experimental Results – Generic Image Deblurring

Blur
SelfDeblur
DeblurGANv2
SRN
Ours
Sharp
Experimental Results – Generic Image Deblurring

| Blur       | SelfDeblur | DeblurGANv2 | SRN   | Ours   | Sharp |
|------------|------------|-------------|-------|--------|-------|
| ![Blur](image1.png) | ![SelfDeblur](image2.png) | ![DeblurGANv2](image3.png) | ![SRN](image4.png) | ![Ours](image5.png) | ![Sharp](image6.png) |
| ![Blur](image7.png) | ![SelfDeblur](image8.png) | ![DeblurGANv2](image9.png) | ![SRN](image10.png) | ![Ours](image11.png) | ![Sharp](image12.png) |
| ![Blur](image13.png) | ![SelfDeblur](image14.png) | ![DeblurGANv2](image15.png) | ![SRN](image16.png) | ![Ours](image17.png) | ![Sharp](image18.png) |
Experimental Results – Blind Image Deblurring

Blur

SelfDeblur

DeblurGANv2 imgaug

DeblurGANv2 REDS

SRN imgaug

SRN REDS

Ours
Experimental Results – Blind Image Deblurring

| Blur | SelfDeblur | DeblurGANv2 | DeblurGANv2 | SRN imgaug | SRN REDS | ours |
|------|------------|-------------|-------------|------------|-----------|------|
| ![Image](blur.png) | ![Image](selfdeblur.png) | ![Image](deblurGANv2.png) | ![Image](deblurGANv2.png) | ![Image](SRN_imgaug.png) | ![Image](SRN_REDS.png) | ![Image](ours.png) |
Experimental Results – Blur Synthesis

Source sharp  Source blur  Synthesized blur
Experimental Results – Blur Synthesis

Source sharp | Source blur | Synthesized blur
Summary

• We have proposed a method to encode the blur kernel space of a deblurring dataset.
• We have proposed some applications of the blur kernel space.