Explore Image Deblurring via Blur Kernel Space

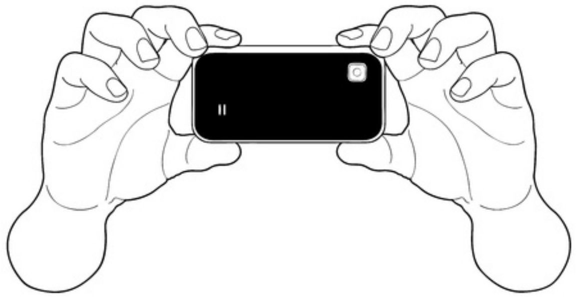
Phong Tran[1] Anh Tran[1, 2] Quynh Phung[1] Minh Hoai[1, 3]

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

https://github.com/VinAIResearch/blur-kernel-space-exploring

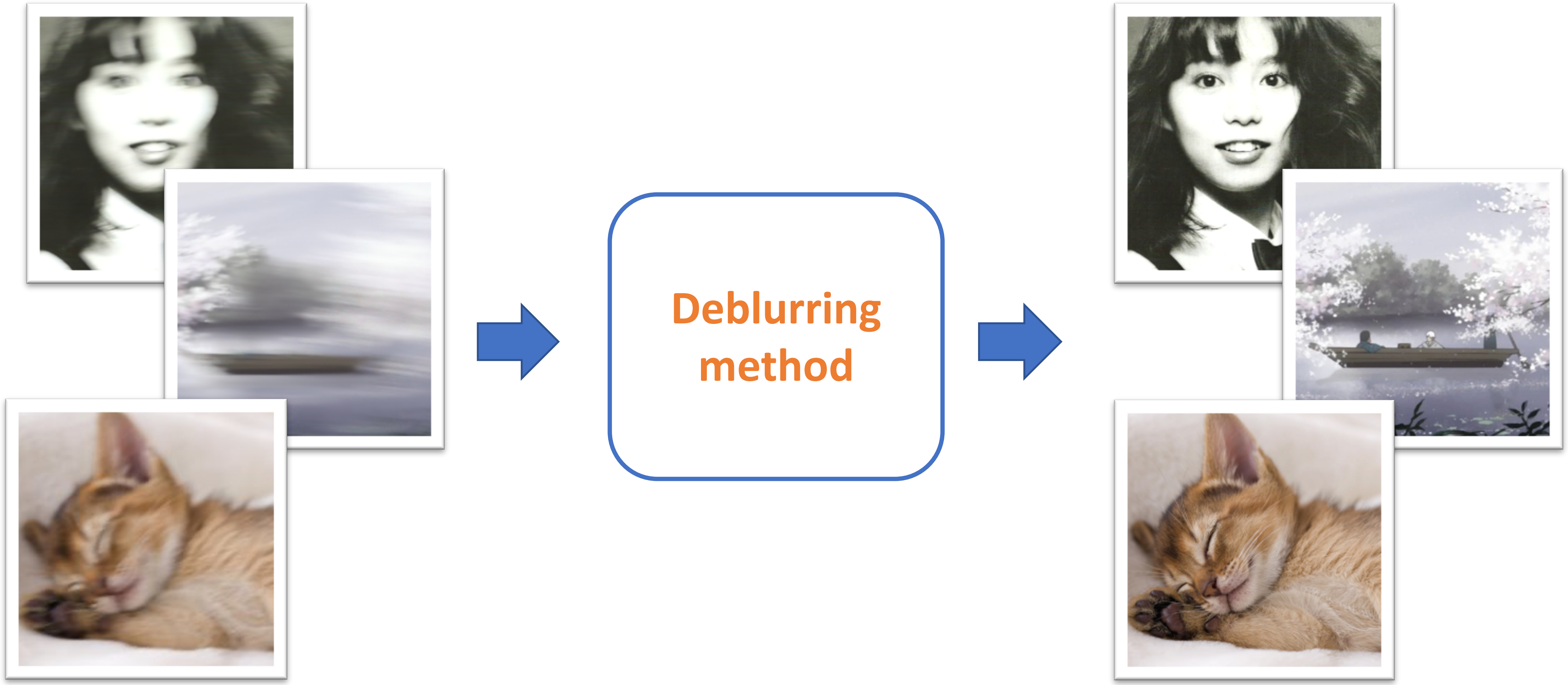
Image Deblurring

Moving object

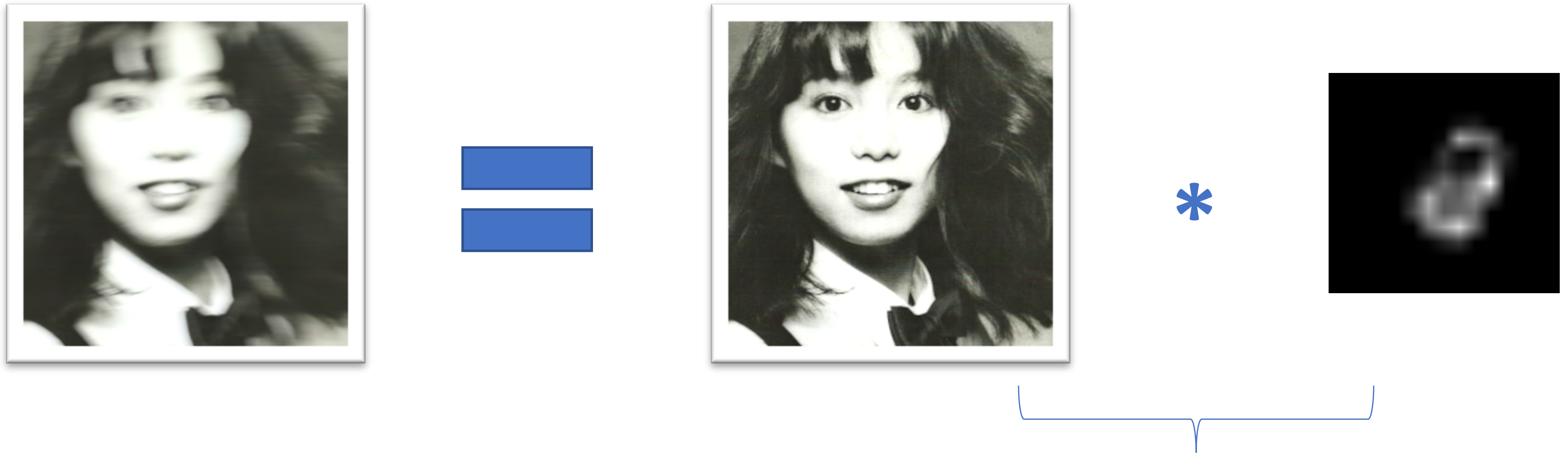


Camera shaking

Image Deblurring



MAP-based Methods



Linear and uniform

$$y = x * k + n$$

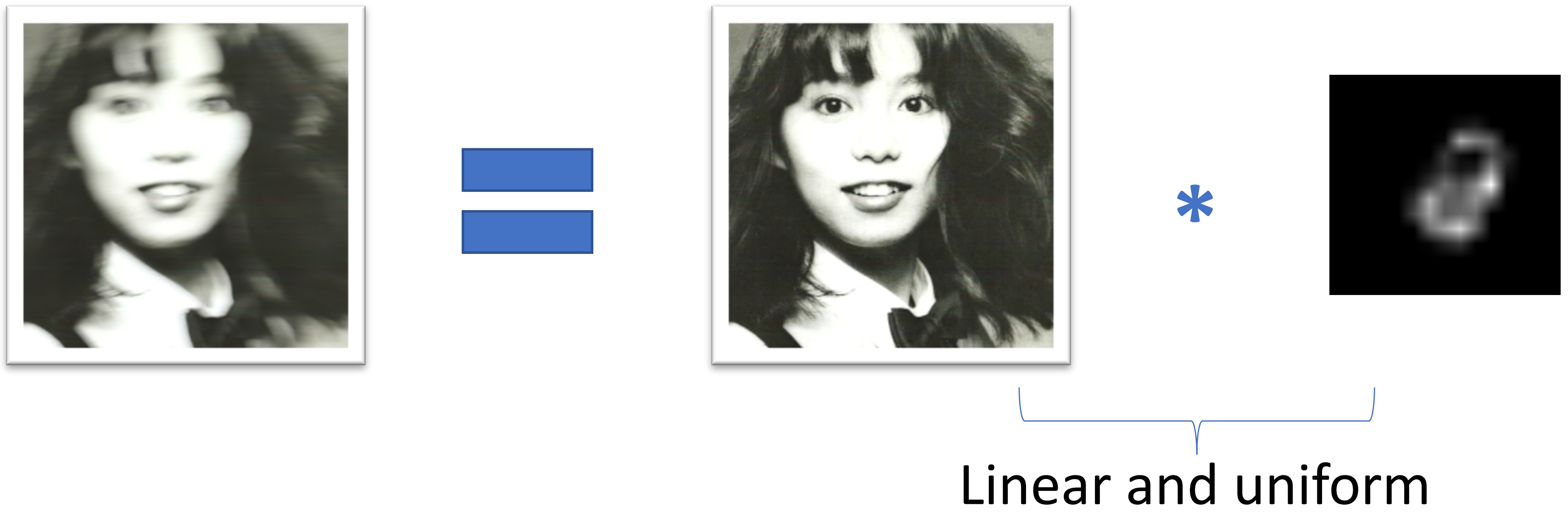
Y: blur image

X: sharp image

K: blur kernel

N: noise

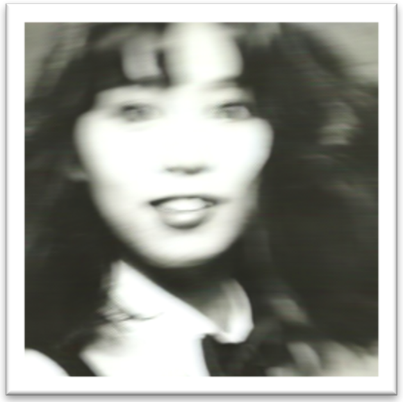
MAP-based Methods



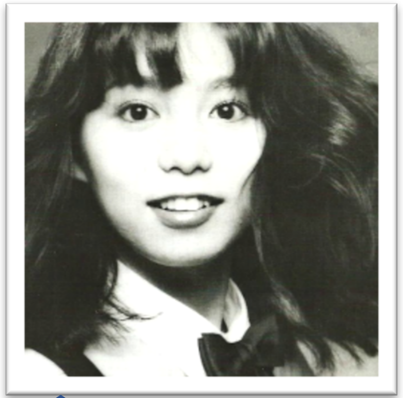
MAP Framework:

$$x, k = \operatorname*{argmax}_{x,k} \mathbb{P}(y|x,k)\mathbb{P}(x)\mathbb{P}(k)$$

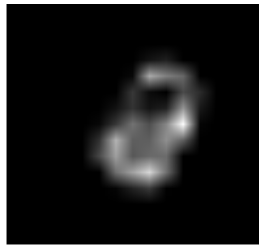
MAP-based Methods



$=$



$*$



Gradient-based penalty, dark channels, ...

Linear and uniform

Sparsity, Spectral properties, ...

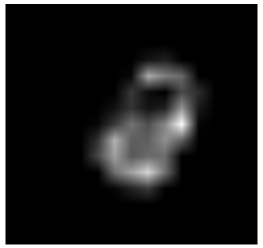
MAP-based Methods

$=$

Does not hold in general

Gradient-based penalty, dark channels, ...

$*$

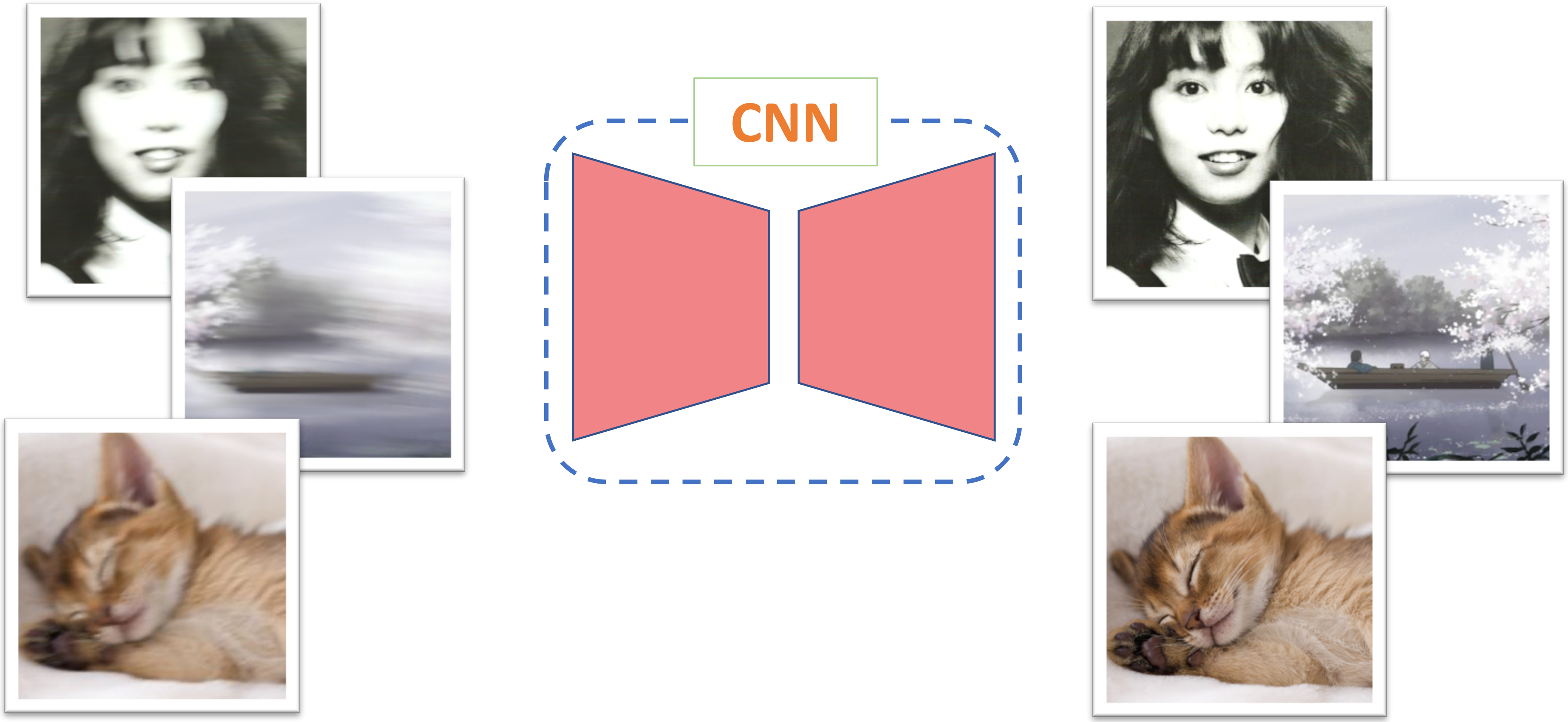


Linear and uniform

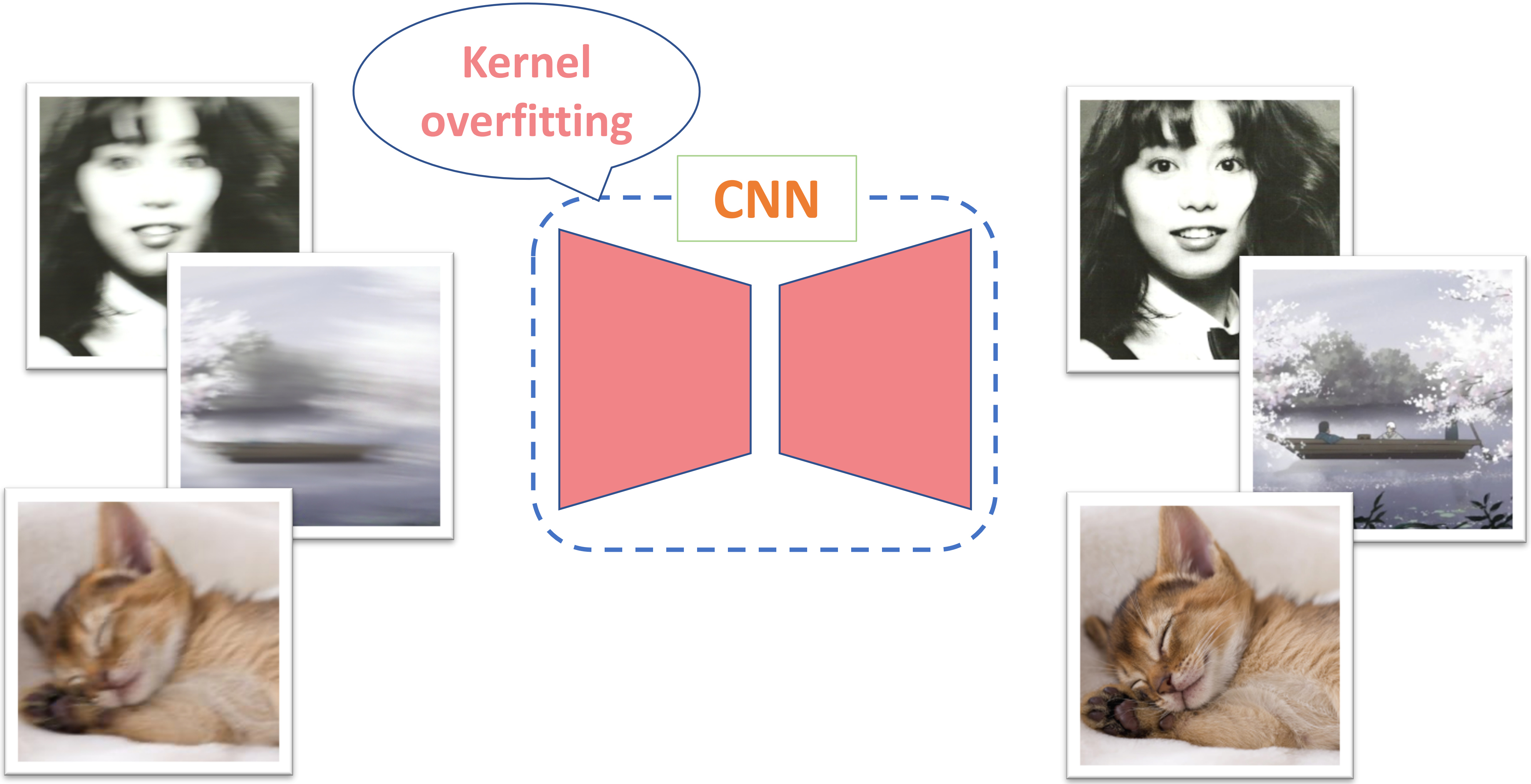
Linear and uniform kernel

Sparsity, Spectral properties, ...

Deep Learning Models



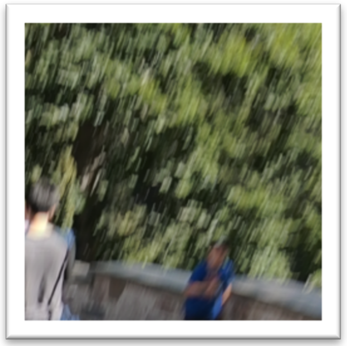
Deep Learning Models - Challenges



Our Work

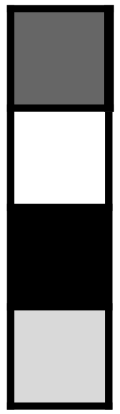
- Generalize MAP-based method
- Leverage neural networks

Our Work



y

x

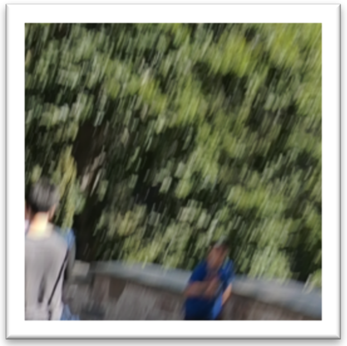


k

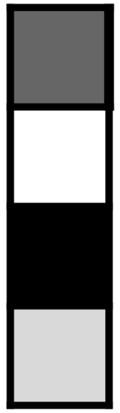
Assumptions: $$\boxed{y = \mathcal{F}(x, k)}$$

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

Our Work


y

x


k

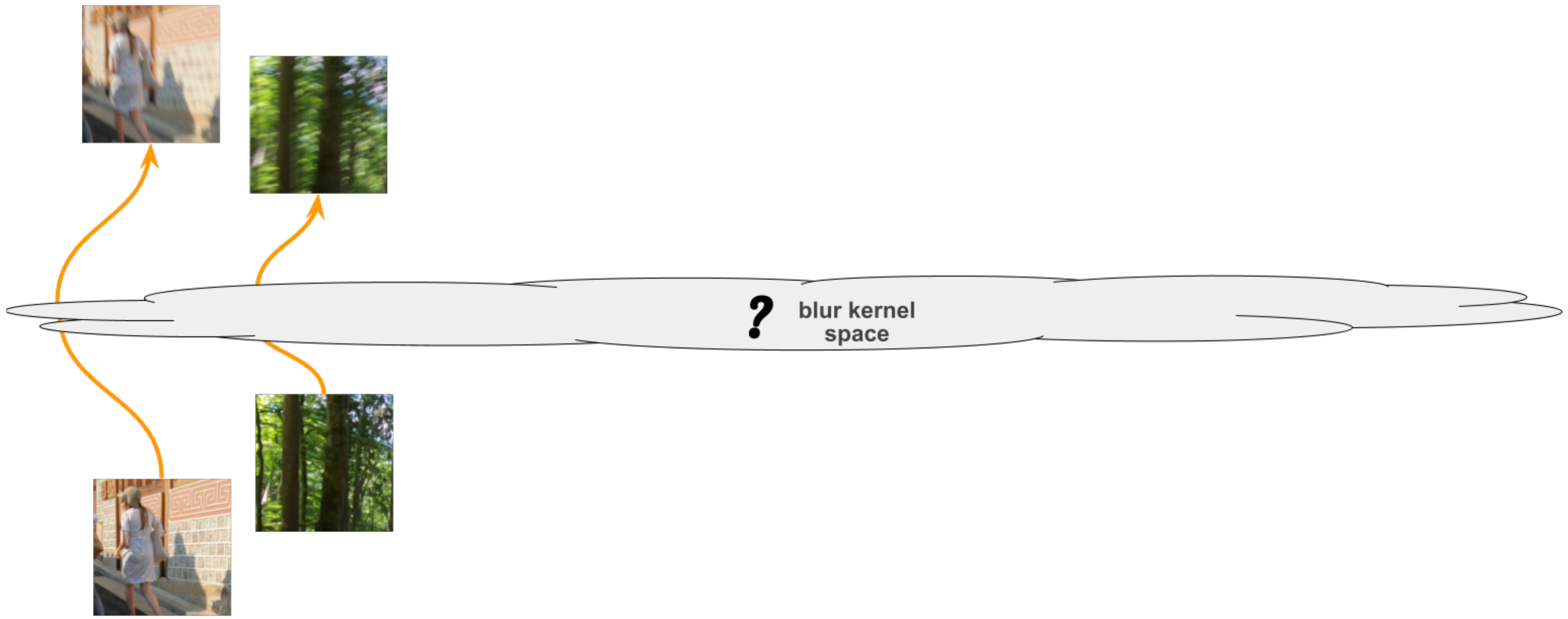
Assumptions: $\boxed{y = \mathcal{F}(x, k)}$

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

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

Our Work

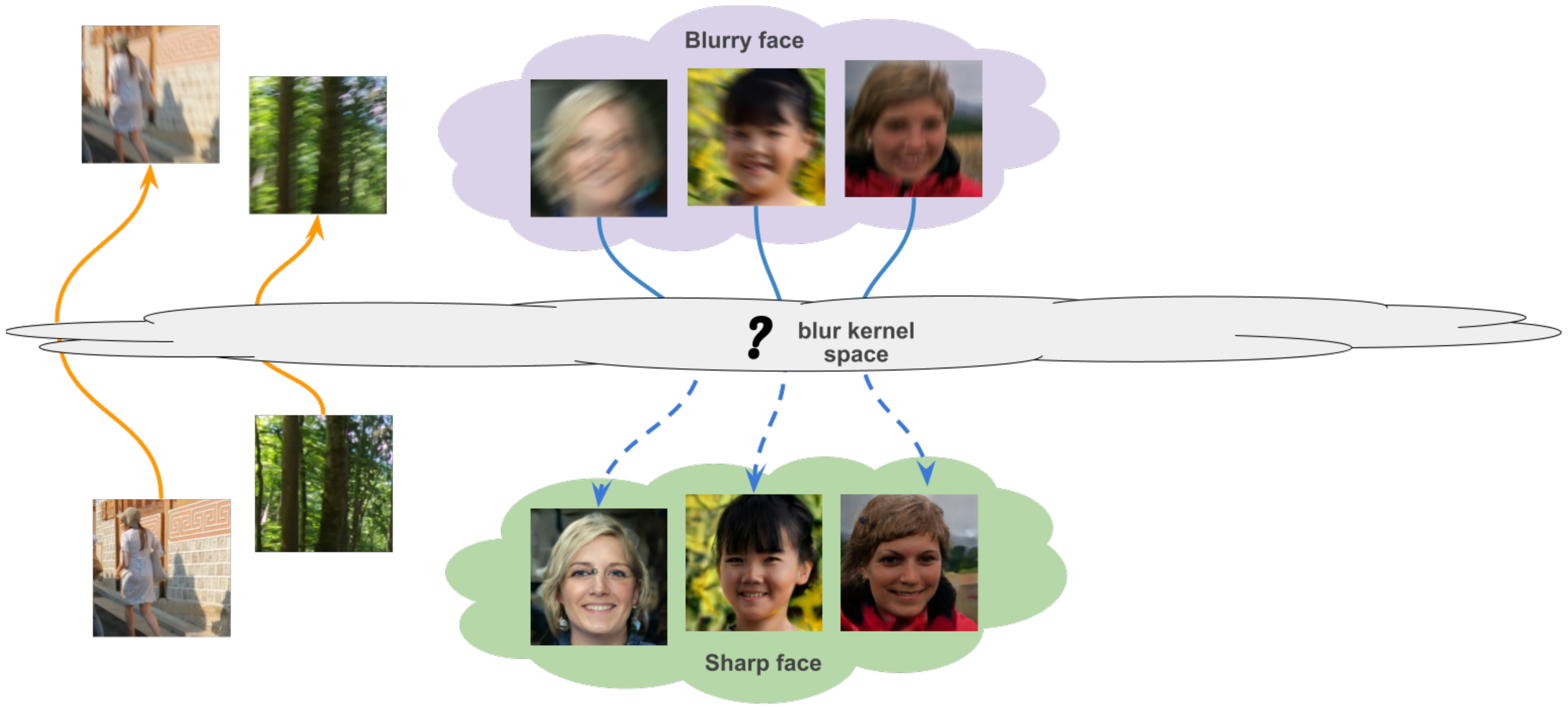
Find F and G



Our Work

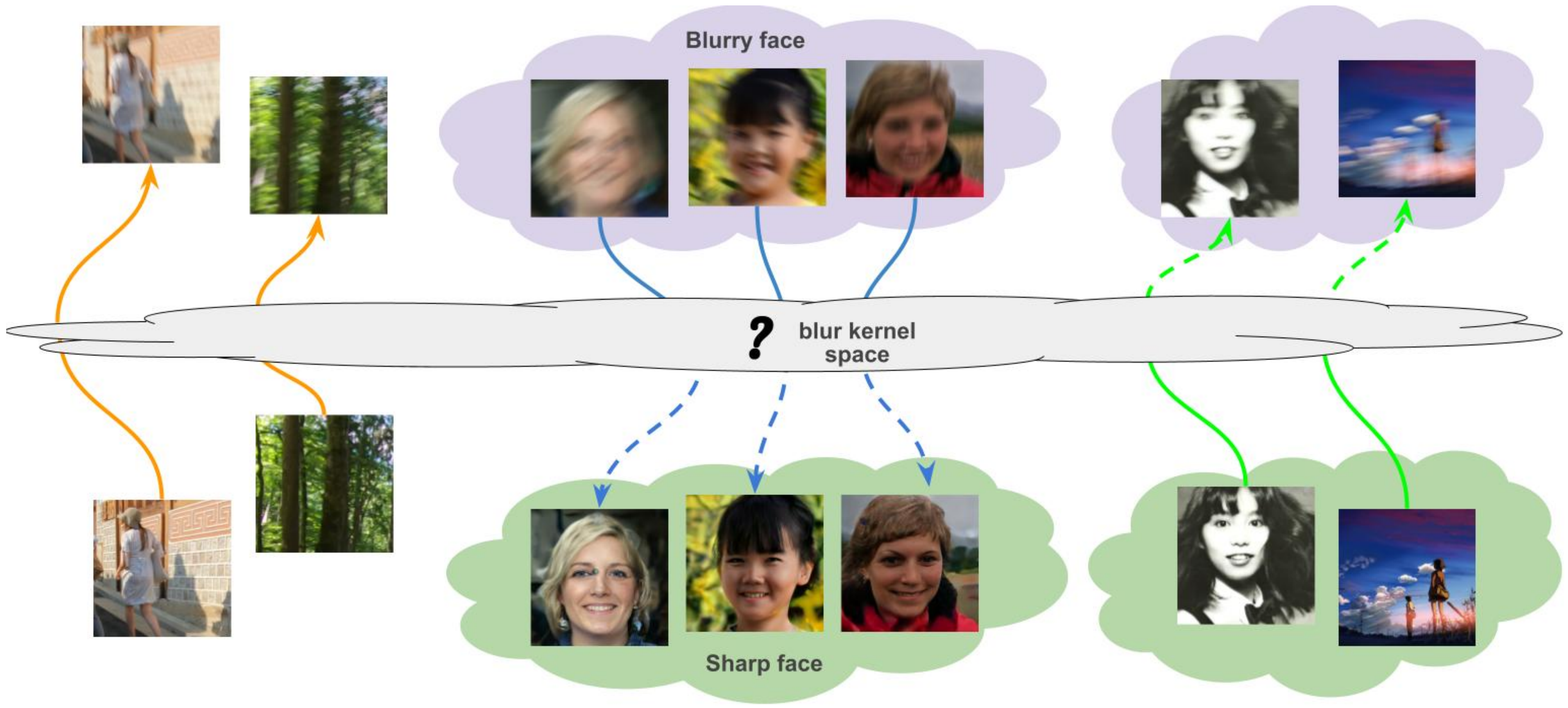
Find F and G

Blind Deblurring



Our Work

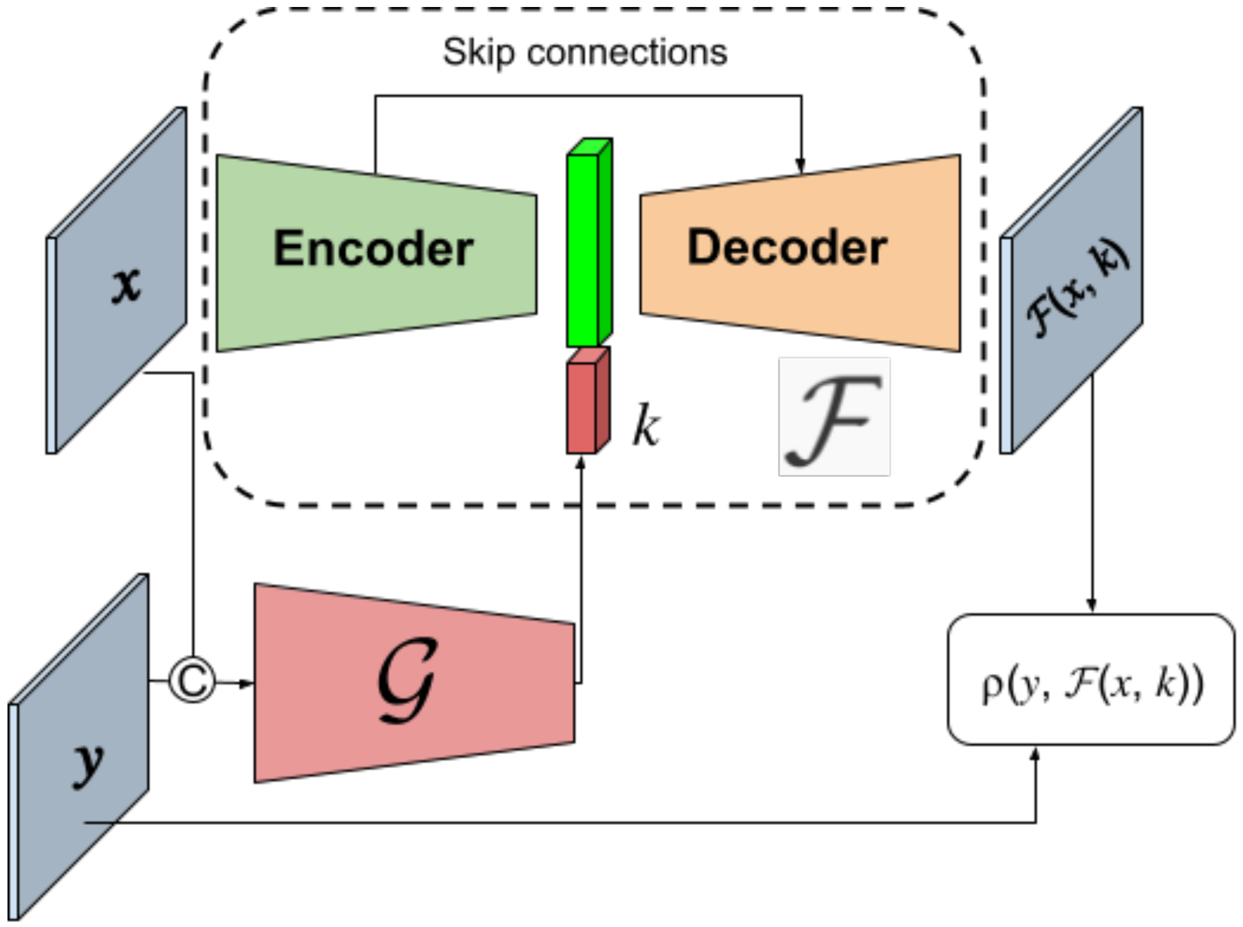
Find F and G   Blind Deblurring   Blur Synthesis



Kernel Encoding

- F and G are implemented by two neural networks.
- For (x, y) ~ $P_{data}(x, y)$. F and G are jointly optimized by minimizing the objective function:

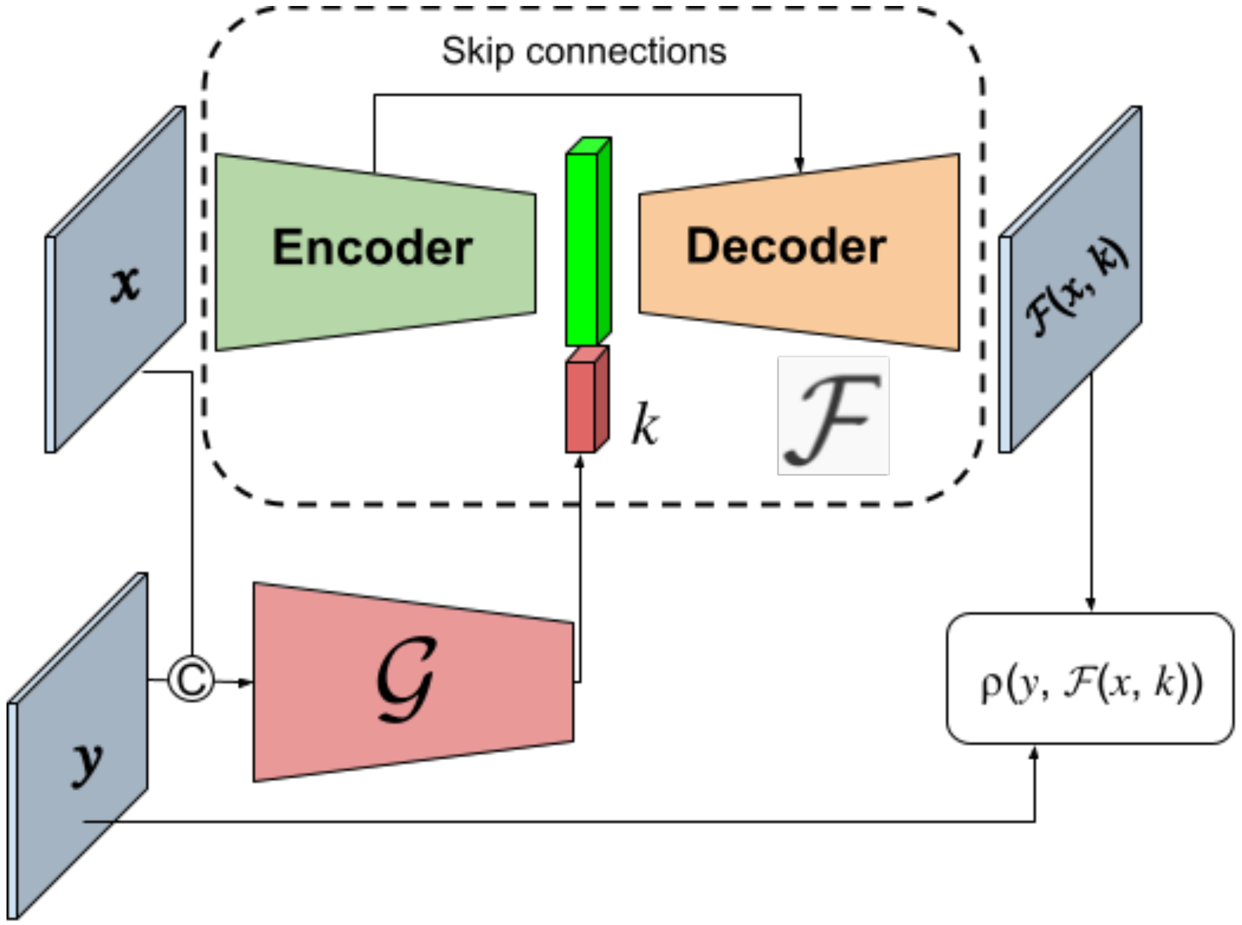
$$\mathbb{E}_{x,y}\left[\rho(y, \mathcal{F}(x, \mathcal{G}(x, y)))\right]$$



Kernel Encoding

- F and G are implemented by two neural networks.
- For (x, y) ~ $P_{data}(x, y)$. F and G are jointly optimized by minimizing the objective function:

$$\underbrace{\mathbb{E}_{x,y}}_{\text{Charbonnier Loss}} \left[\rho(y, \underbrace{\mathcal{F}(x, \mathcal{G}(x, y))}_{\text{Recon blurry image}})\right]$$



Generic Image Deblurring

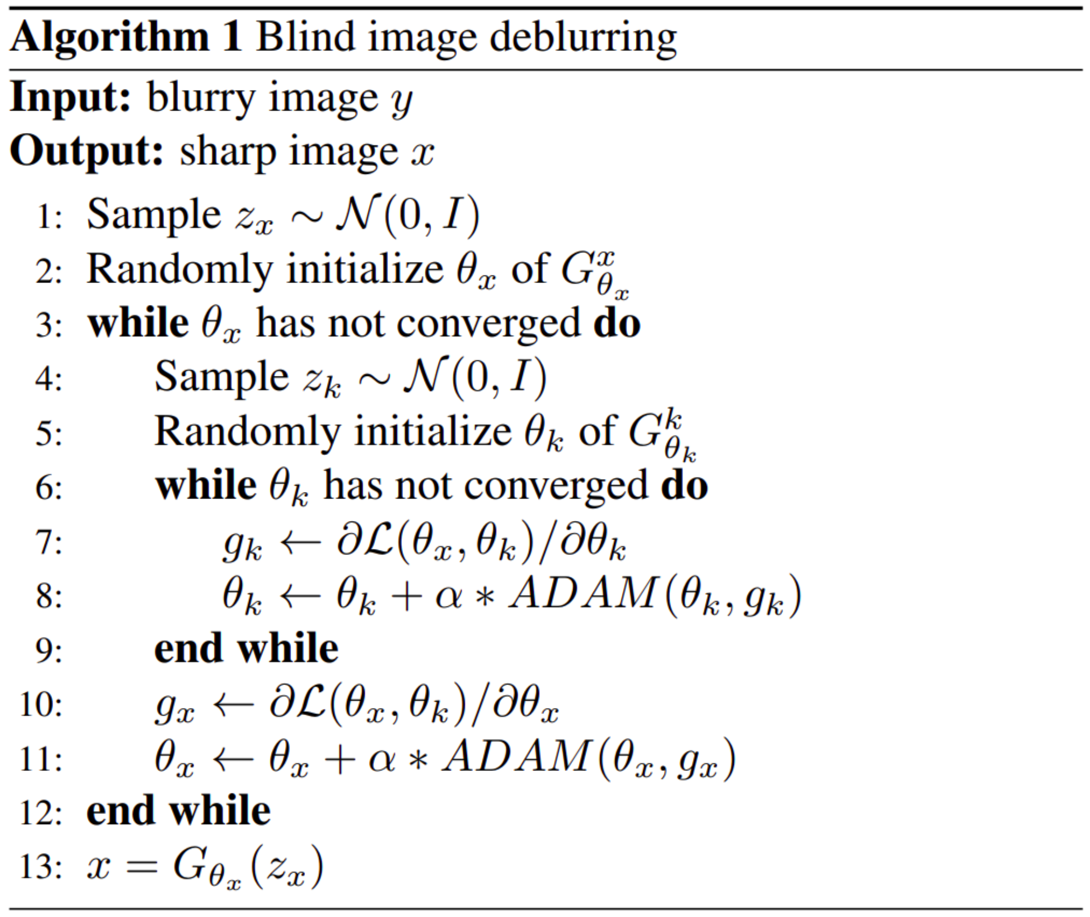
- X and k are alternatively optimized by minimizing:

$$\sum_{i=1}^{n} \underbrace{\rho}_{\text{Charbonnier Loss}}(y_i, \underbrace{\mathcal{F}(x_i, \mathcal{G}(x_i, y_i))}_{\text{Recon blurry image}})$$

Generic Image Deblurring

- X and k are alternatively optimized by minimizing:

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



Algorithm 1 Blind image deblurring

Input: blurry image y

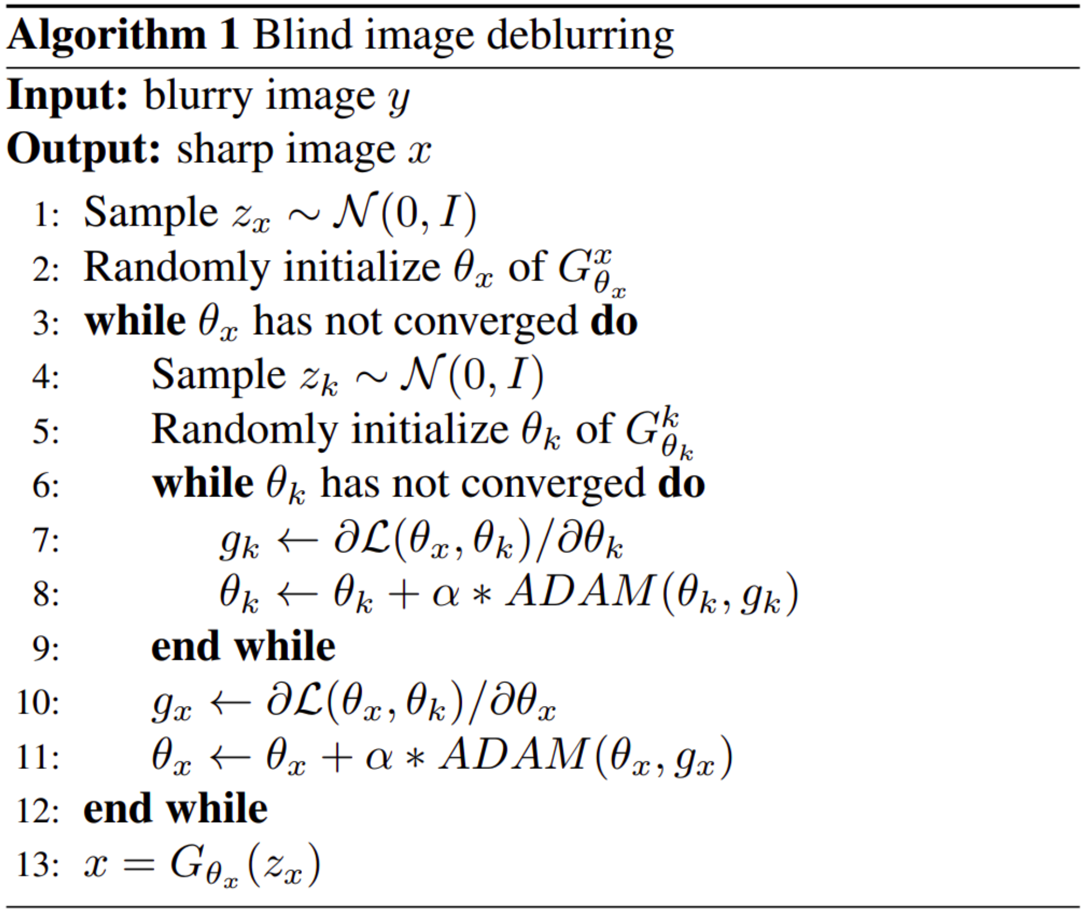
Output: sharp image x

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

Generic Image Deblurring

- X and k are alternatively optimized by minimizing:

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



Algorithm 1 Blind image deblurring

Input: blurry image y

Output: sharp image x

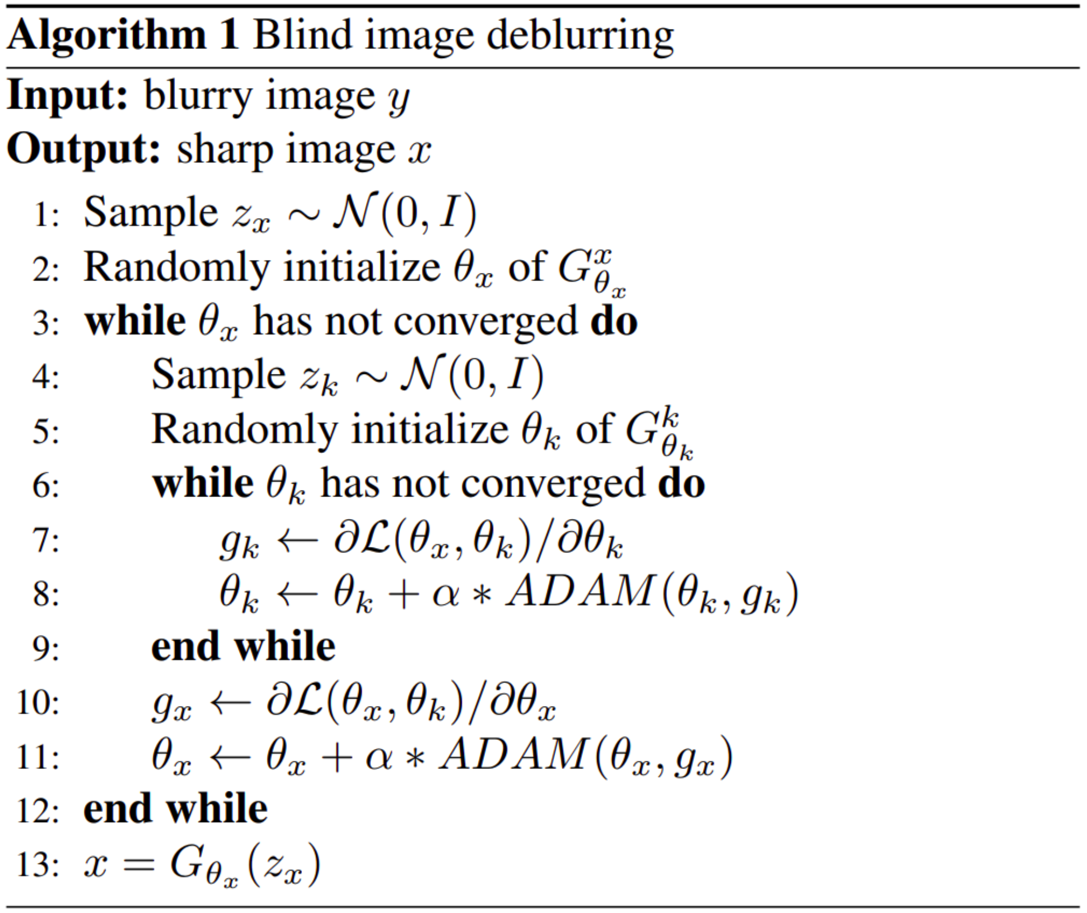
```
1: Sample $z_x \sim \mathcal{N}(0, I)$
2: Randomly initialize $\theta_x$ of $G^x_{\theta_x}$
3: while $\theta_x$ has not converged do
4:     Sample $z_k \sim \mathcal{N}(0, I)$
5:     Randomly initialize $\theta_k$ of $G^k_{\theta_k}$
6:     while $\theta_k$ has not converged do
7:         $g_k \leftarrow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

fix k, optimize x

Generic Image Deblurring

- X and k are alternatively optimized by minimizing:

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



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

fix x, optimize k

fix k, optimize x

Generic Image Deblurring

- X and k are alternatively optimized by minimizing:

$$\rho(y, \mathcal{F}(x,k)) + \underbrace{\lambda||k||_2 + \gamma(g_u^2(x) + g_v^2(x))^{\alpha/2}}_{\text{Regularization term}}$$

fix x, optimize k

fix k, optimize x

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

Generic Image Deblurring

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

- X and k are alternatively optimized by minimizing:

$$\rho(y, \mathcal{F}(x, k)) + \underbrace{\lambda ||k||_2 + \gamma(g_u^2(x) + g_v^2(x))^{\alpha/2}}_{\text{Regularization term}}$$

DIP x

DIP k

fix x, optimize k

fix k, optimize x

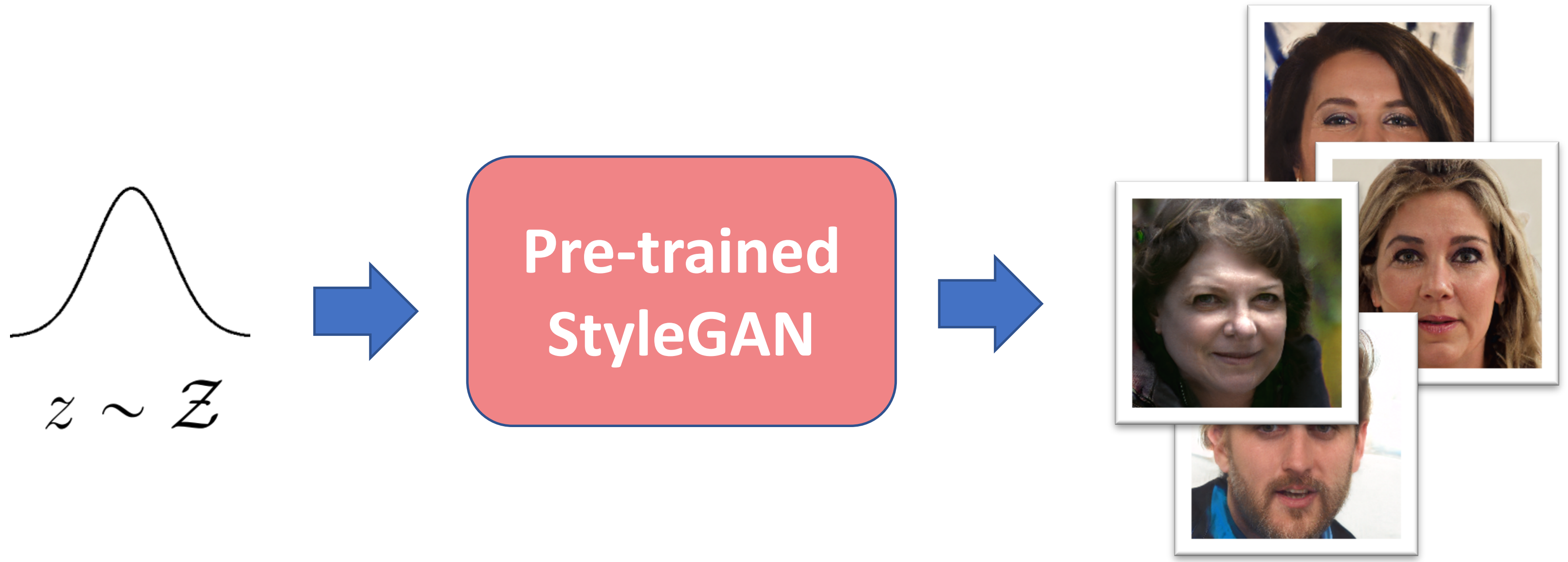
Algorithm 1 Blind image deblurring

Input: blurry image y

Output: sharp image x

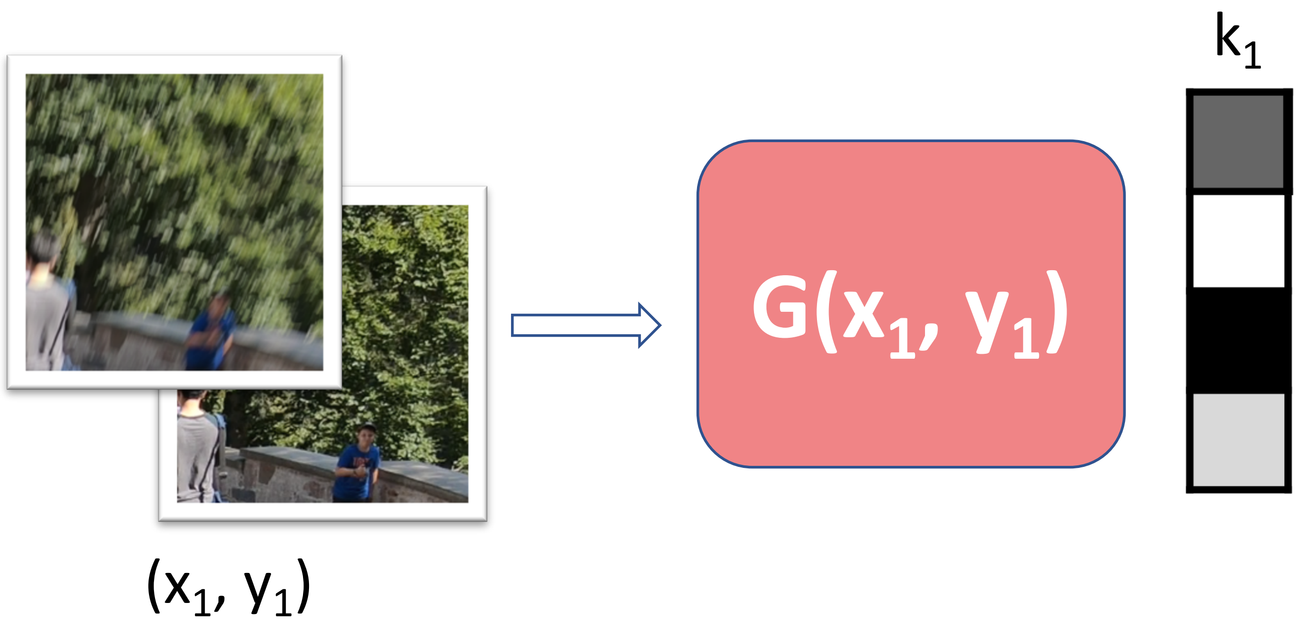
```
1:  Sample $z_x \sim \mathcal{N}(0, I)$
2:  Randomly initialize $\theta_x$ of $G^x_{\theta_x}$
3:  **while** $\theta_x$ has not converged **do**
4:      Sample $z_k \sim \mathcal{N}(0, I)$
5:      Randomly initialize $\theta_k$ of $G^k_{\theta_k}$
6:      **while** $\theta_k$ has not converged **do**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

Domain-specific Image Deblurring

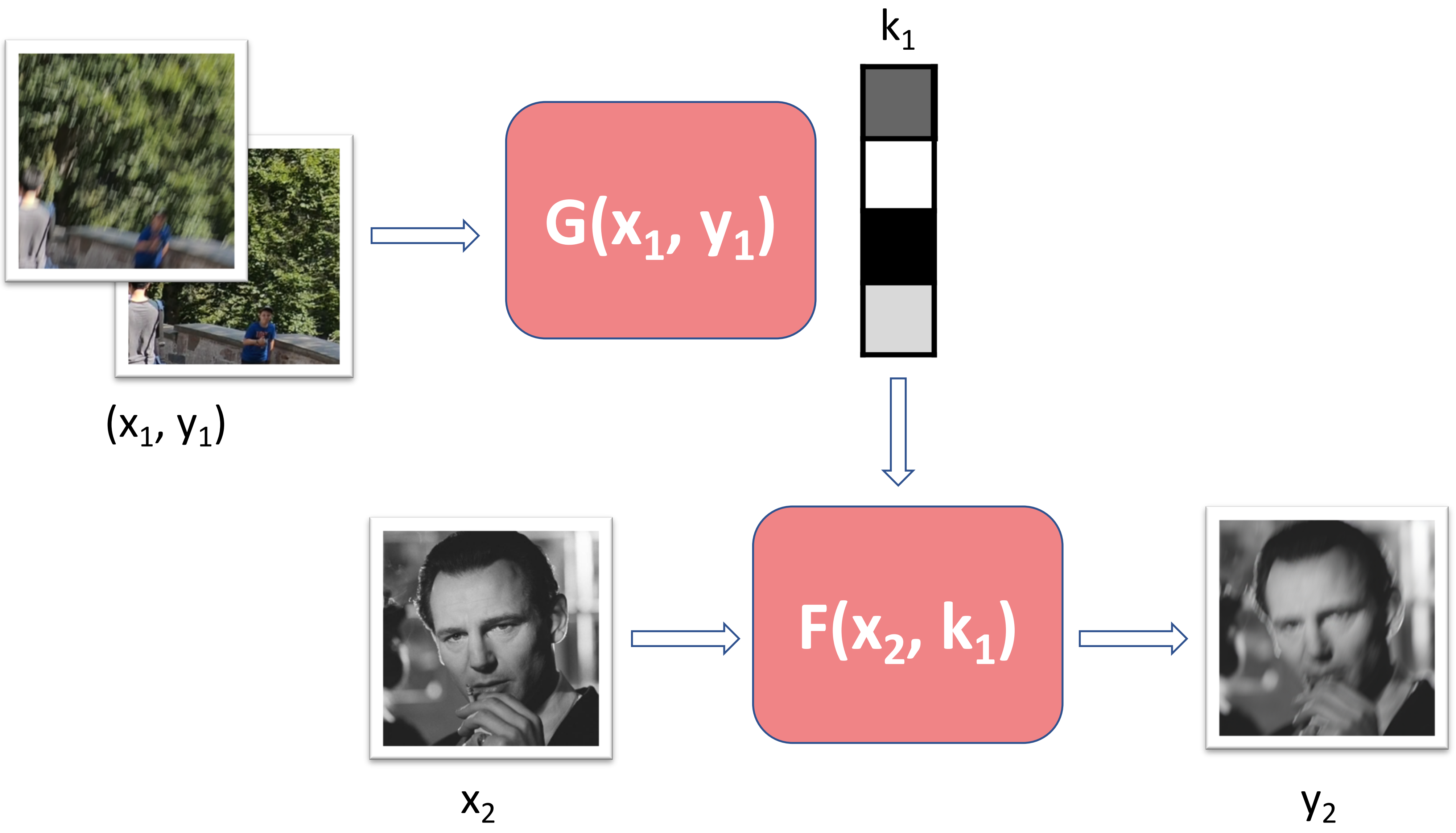


$$z^*, k^* = \arg\max_{z,k} \rho\left(\mathcal{F}(G_{style}(z), k), y)\right) + \underbrace{R_z(z) + R_k(k)}_{\textbf{Regularization term}}$$

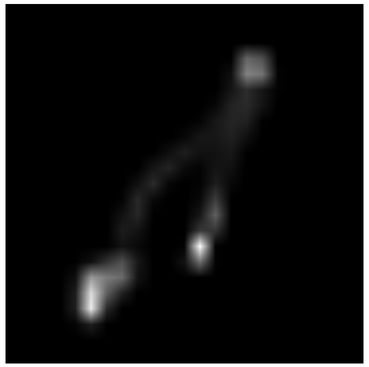
Blur Synthesis

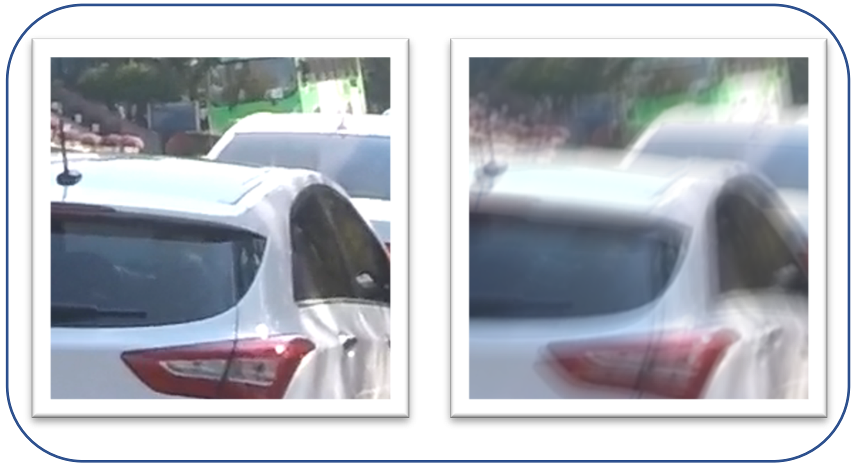


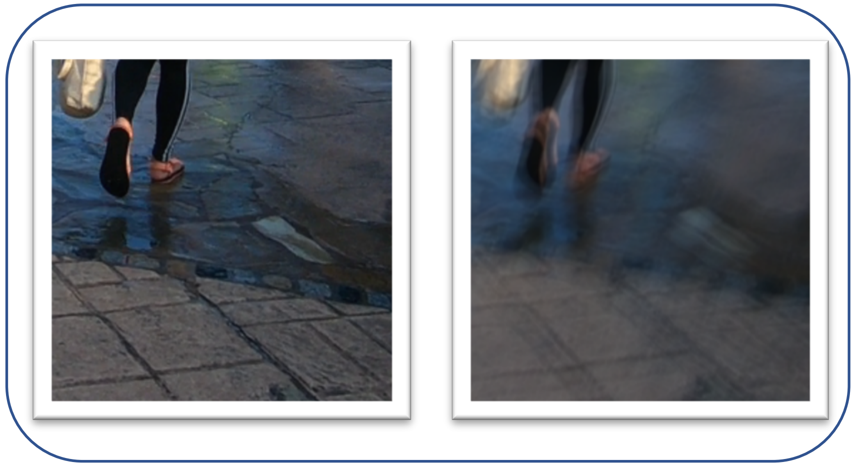
Blur Synthesis



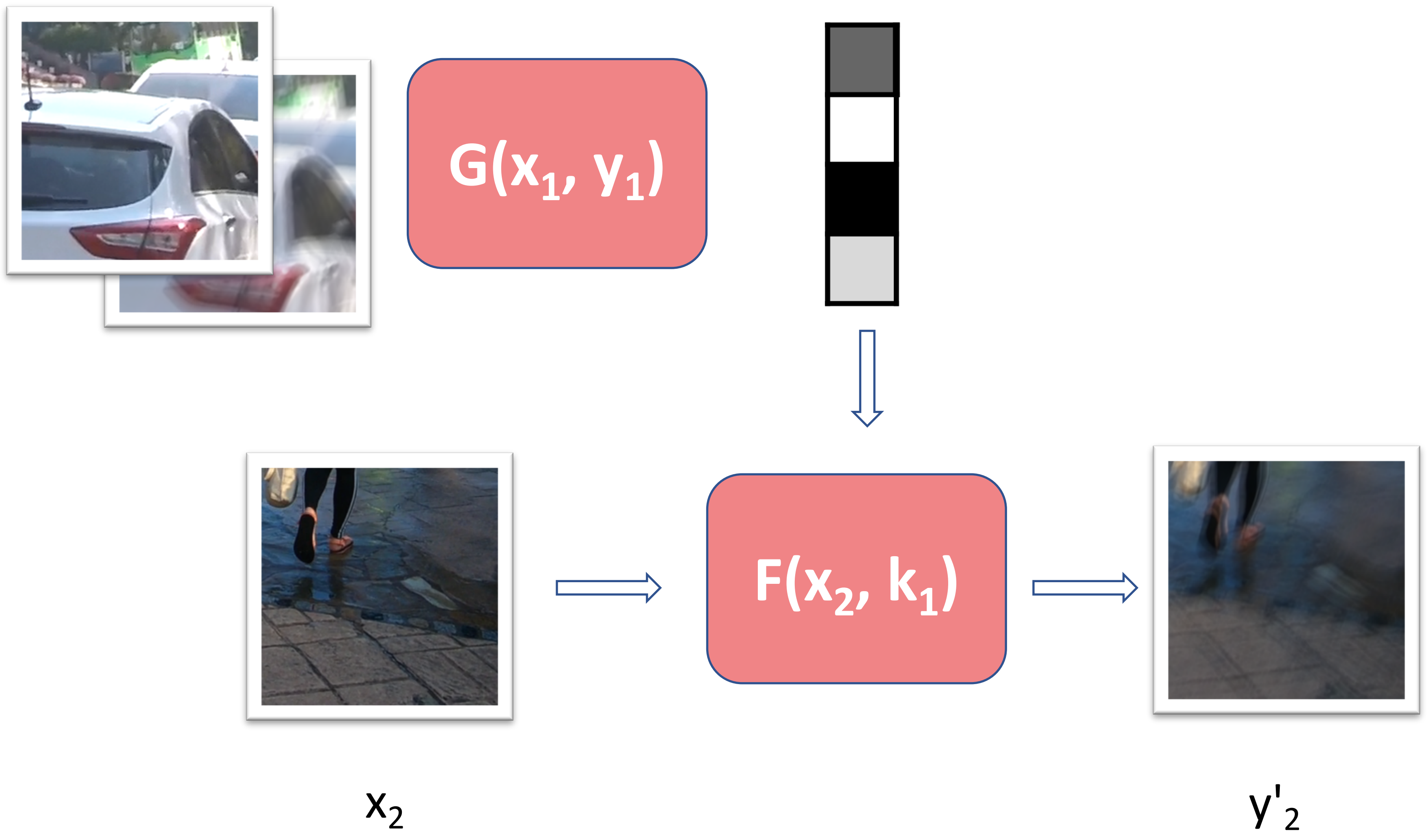
Experimental Results – Kernel Encoding


Kernel 8

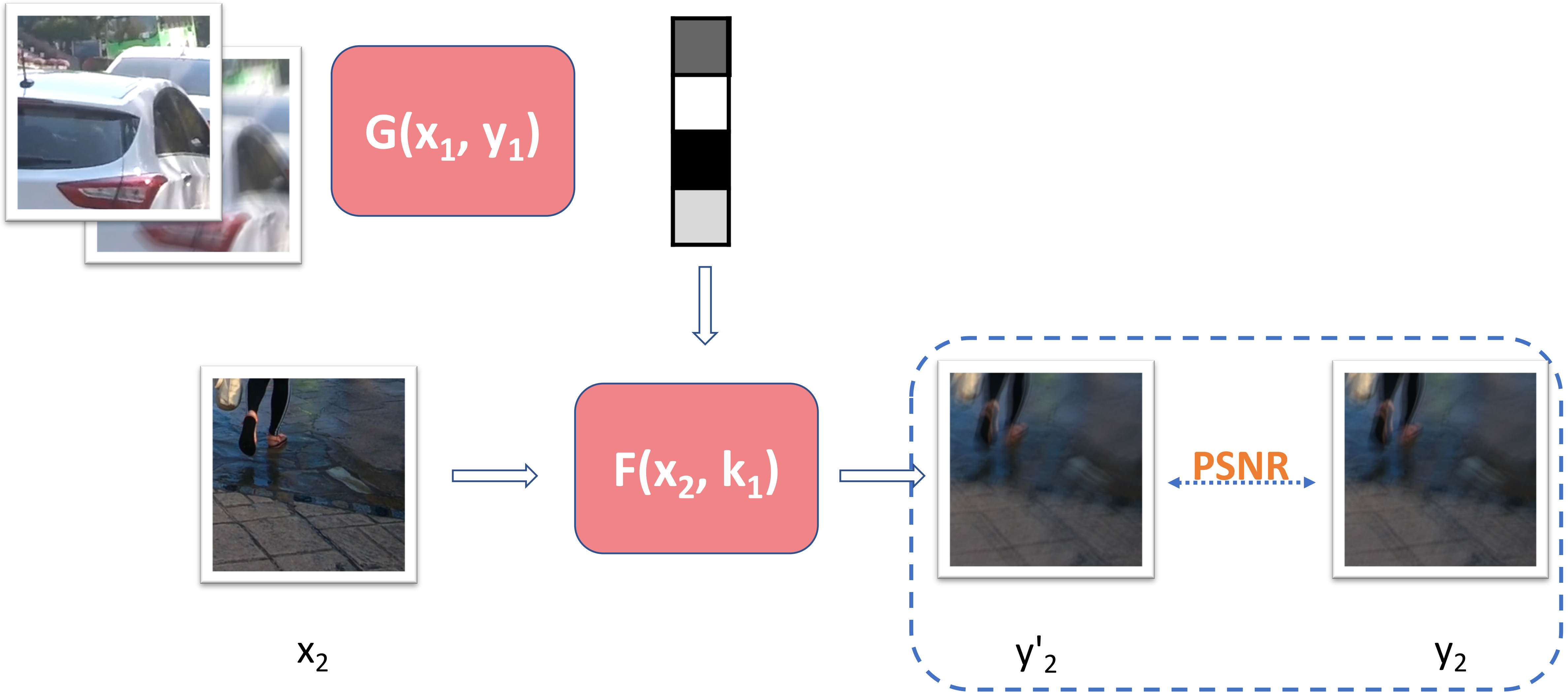

(x_1, y_1)


(x_2, y_2)

Experimental Results – Kernel Encoding



Experimental Results – Kernel Encoding

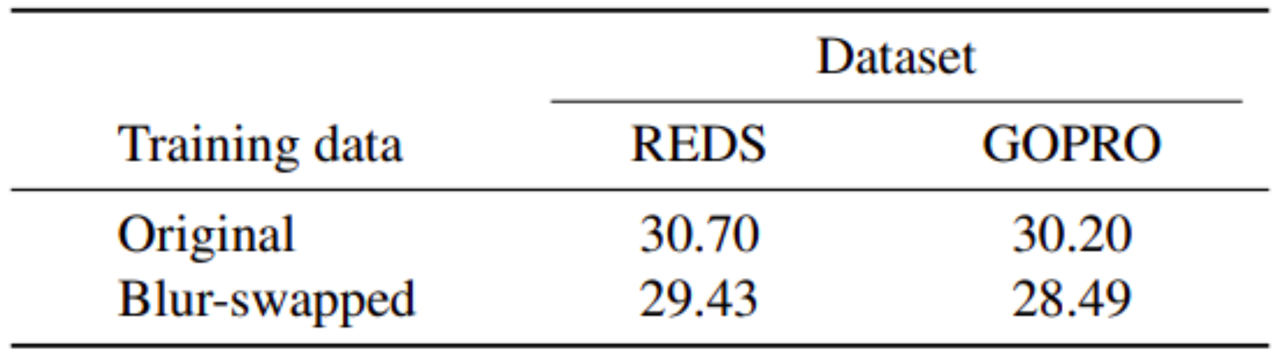


Experimental Results – Kernel Encoding

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

Blur transferring performance on Levin dataset

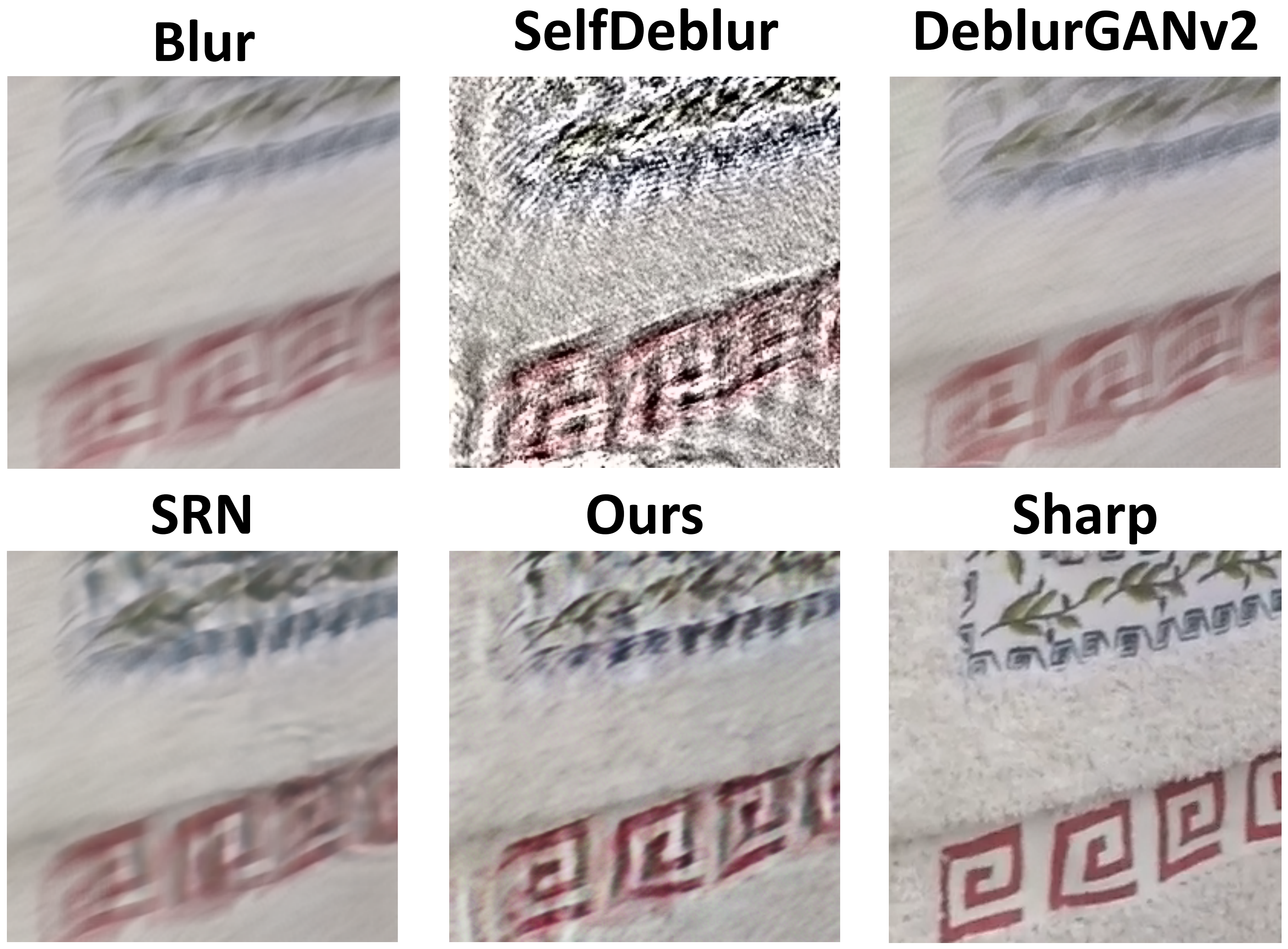
Experimental Results – Kernel Encoding



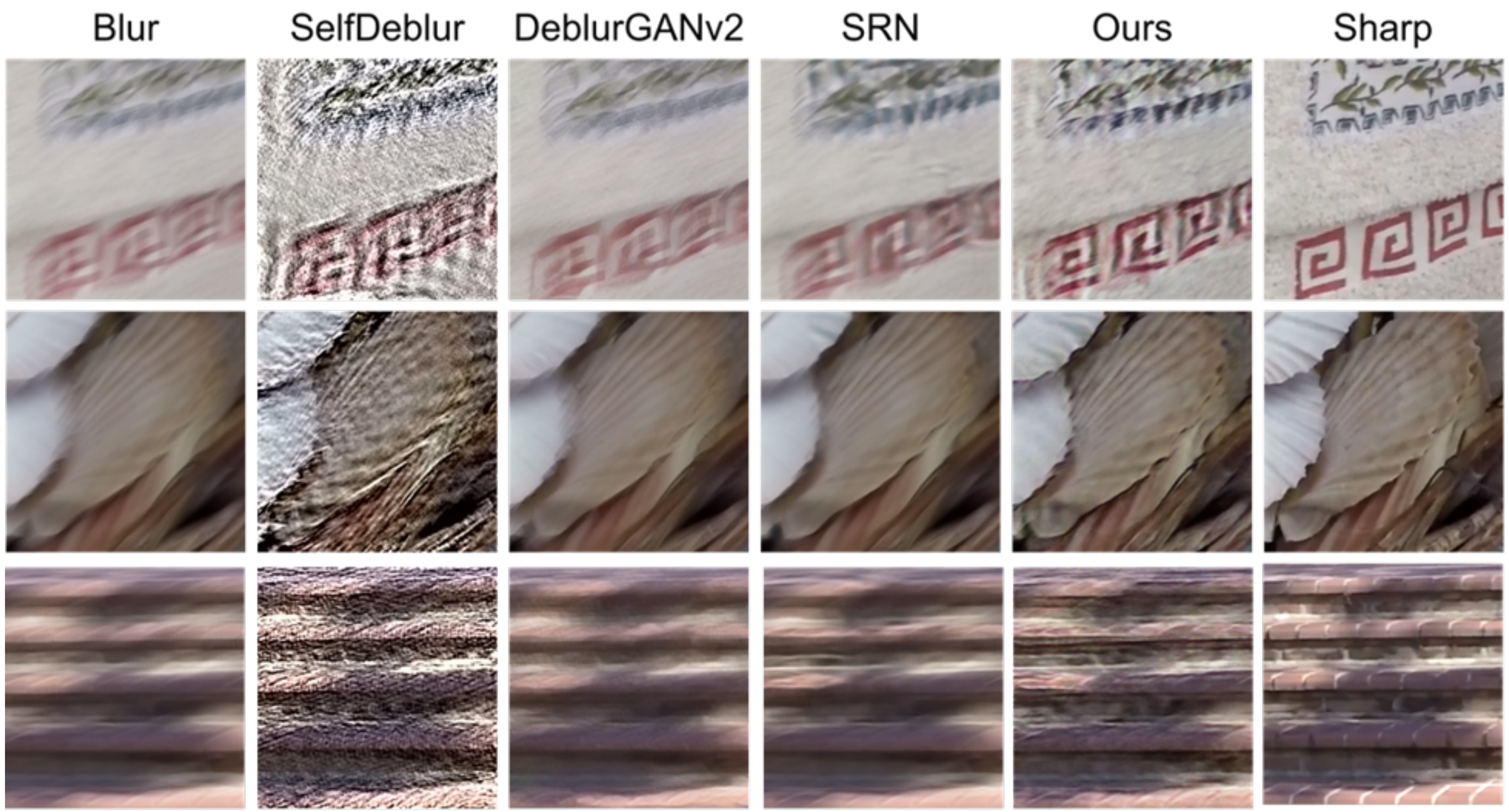
Training data	Dataset	
	REDS	GOPRO
Original	30.70	30.20
Blur-swapped	29.43	28.49

SRN performance when training on blur-swapped dataset

Experimental Results – Generic Image Deblurring

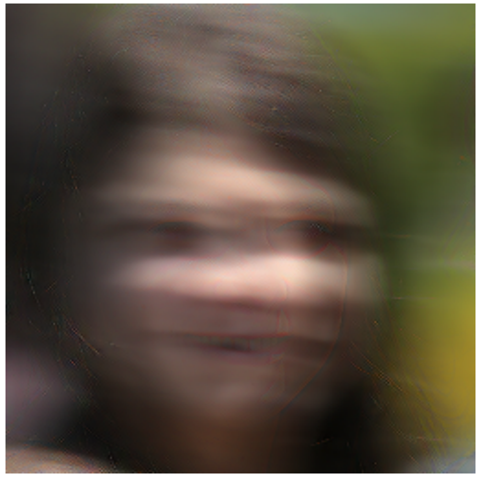


Experimental Results – Generic Image Deblurring



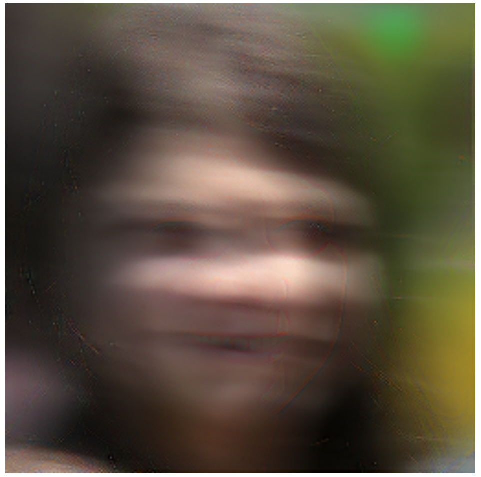
Experimental Results – Blind Image Deblurring

Blur

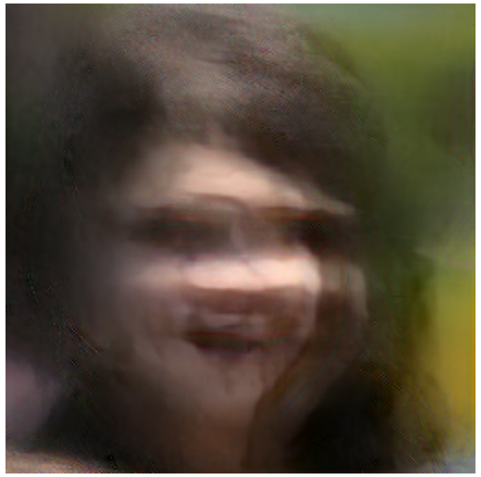


SelfDeblur

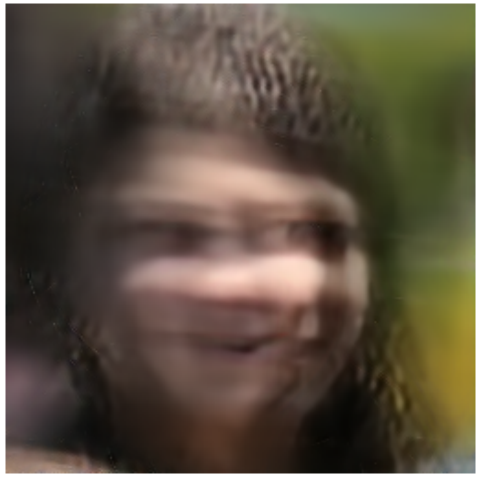
DeblurGANv2 imgaug



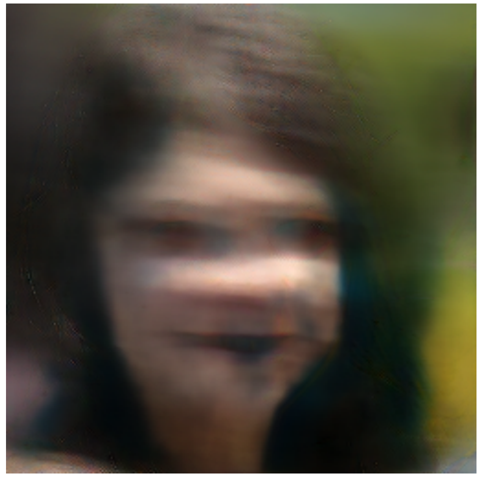
DeblurGANv2 REDS



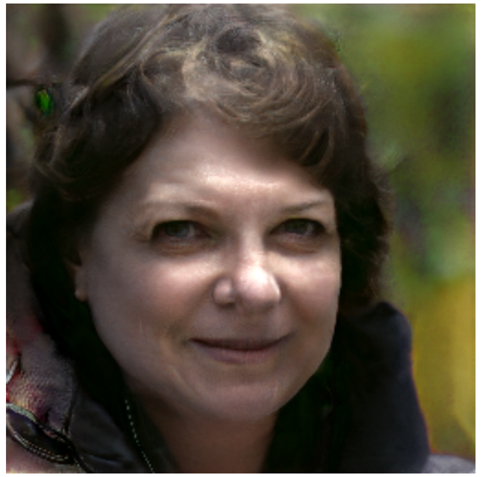
SRN imgaug



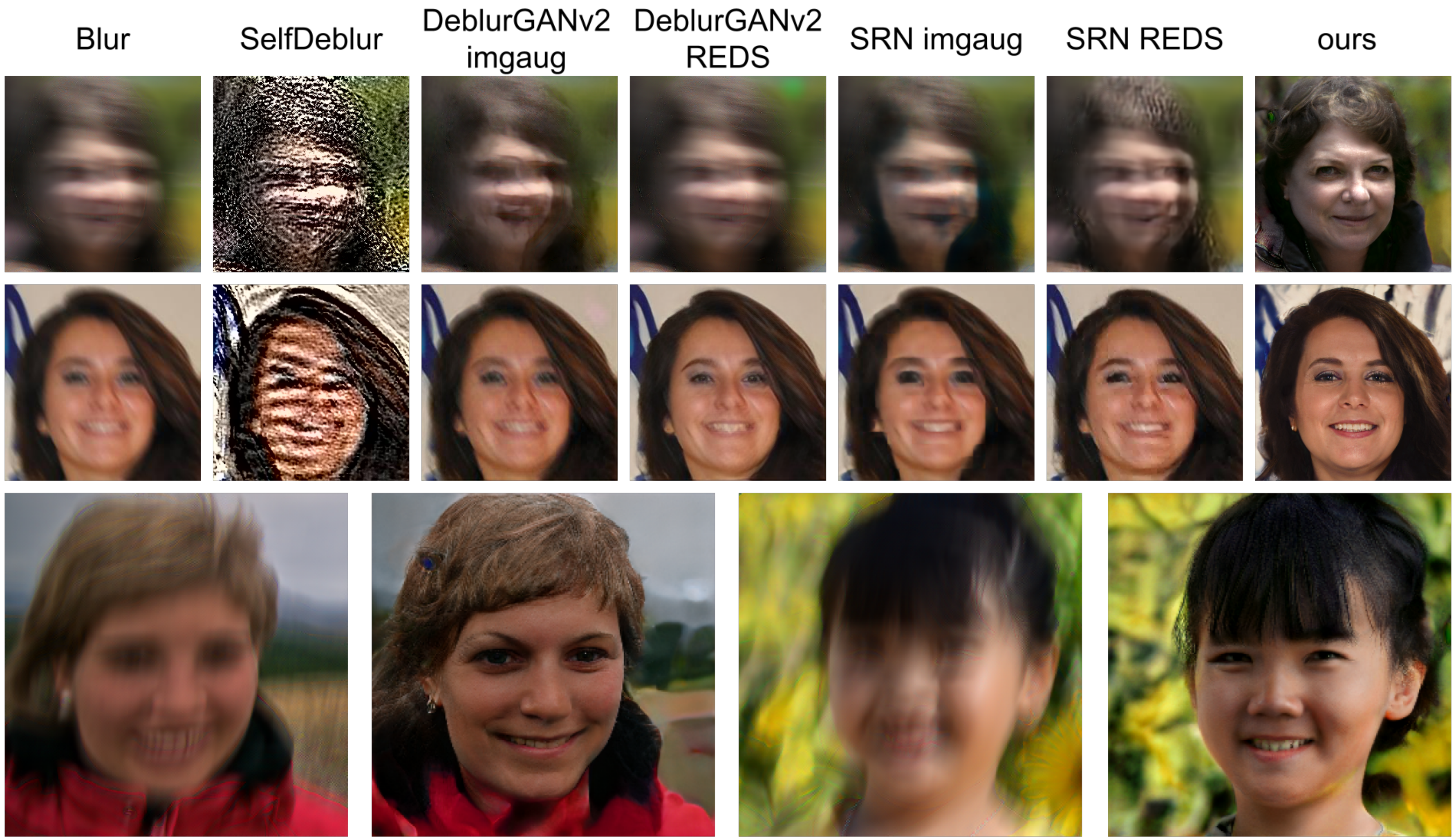
SRN REDS



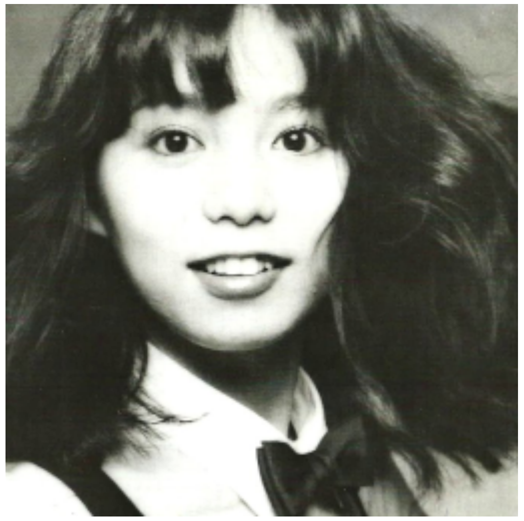
Ours



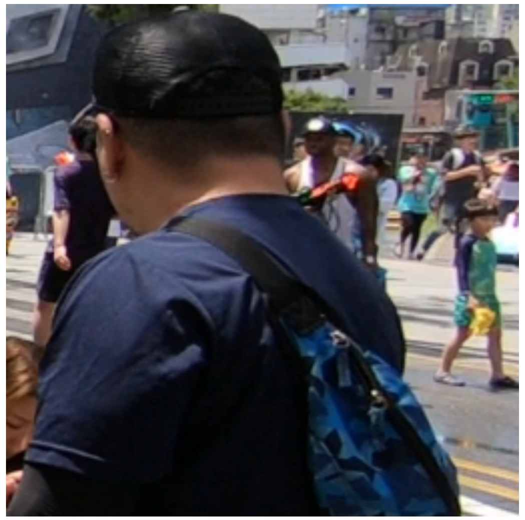
Experimental Results – Blind Image Deblurring



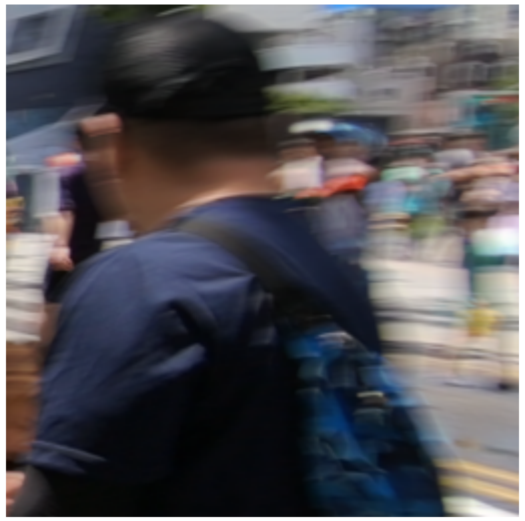
Experimental Results – Blur Synthesis



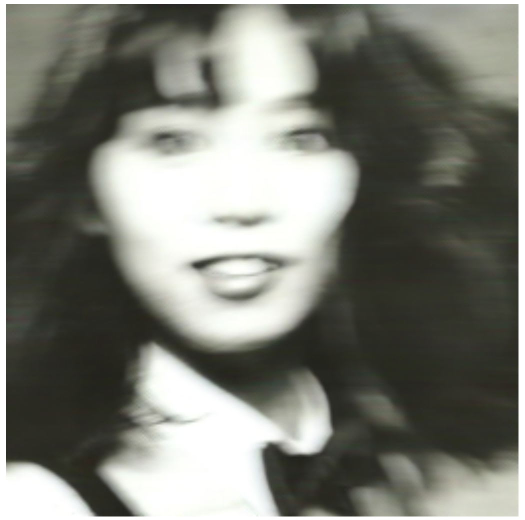
Source sharp



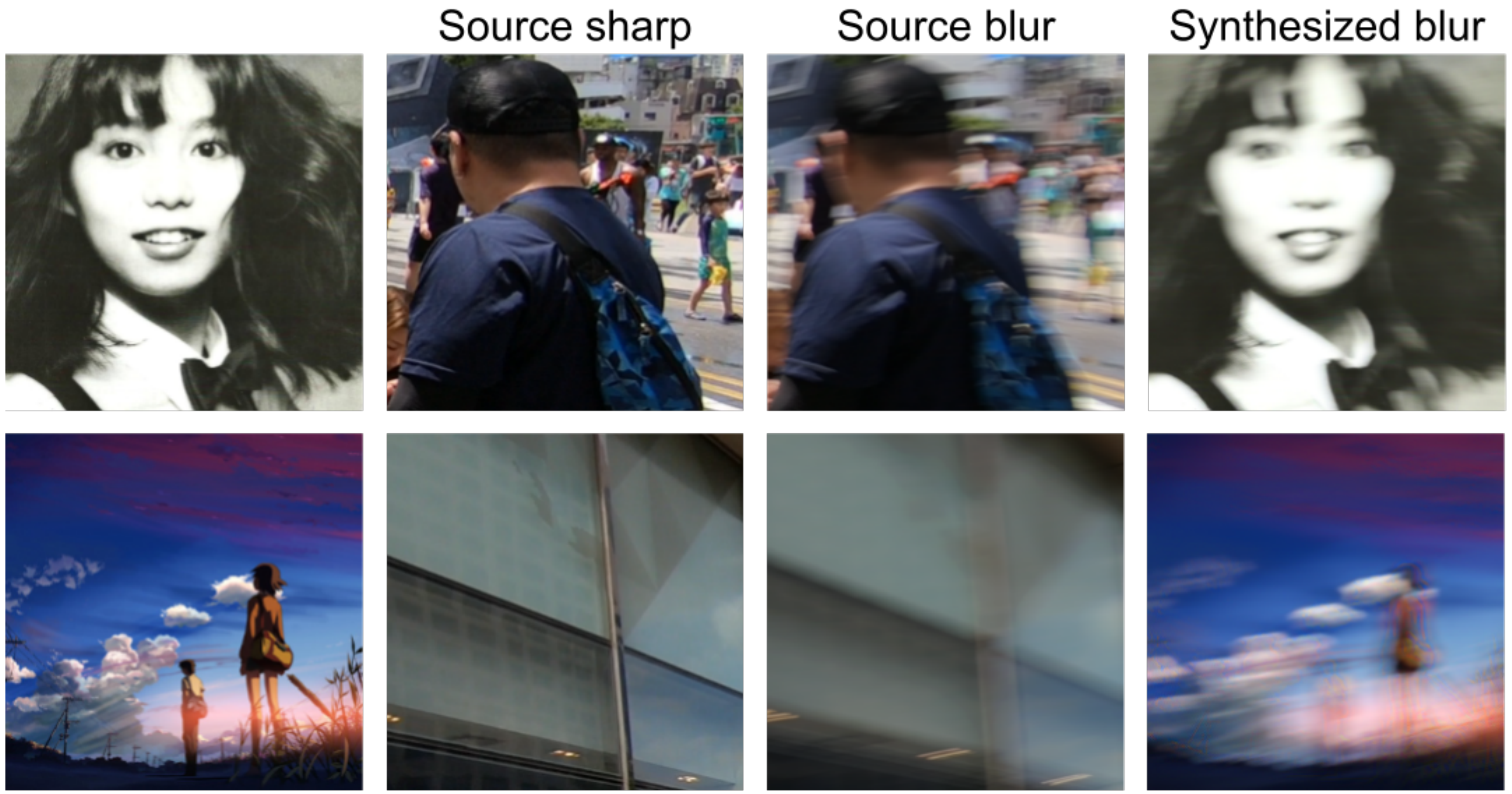
Source blur



Synthesized blur



Experimental Results – Blur Synthesis



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.

Code

https://github.com/VinAIResearch/blur-kernel-space-exploring

Paper

https://www.vinai.io/publication-posts/explore-image-deblurring-via-encoded-blur-kernel-space/