Exploring Blur Kernel Space

2021 연구실 하계 세미나

민 성 준

Vision & Display Systems Lab.

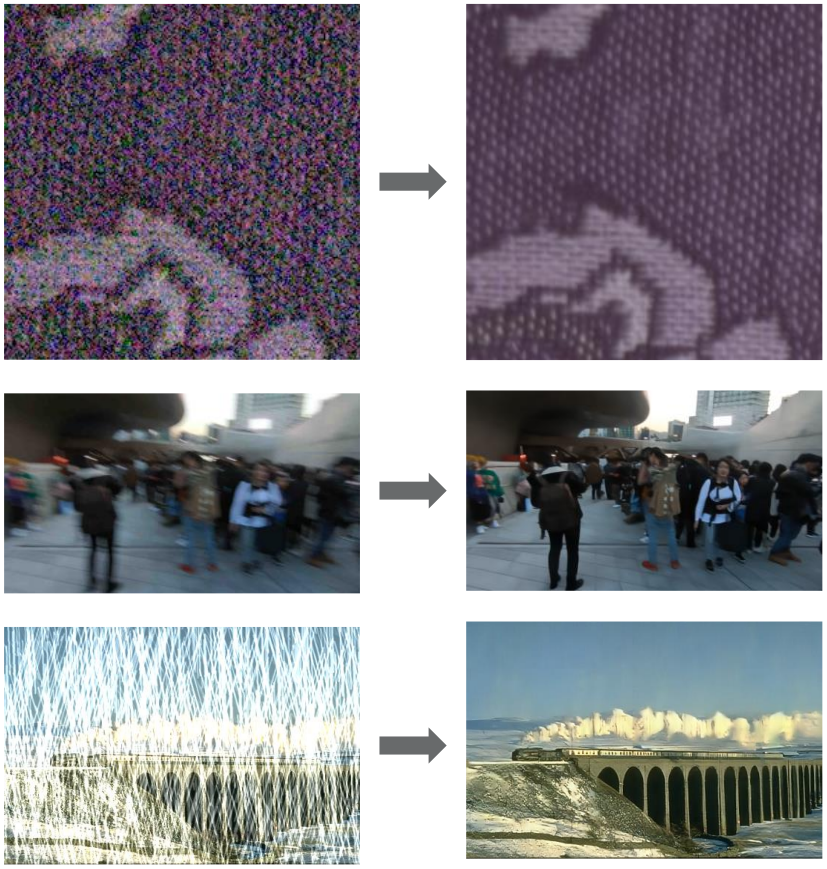
Dept. of Electronic Engineering, Sogang University

Outline

- Introduction
- Background
- Explore Image Deblurring via Encoded Blur Kernel Space
- Experiments
- Conclusion
- References

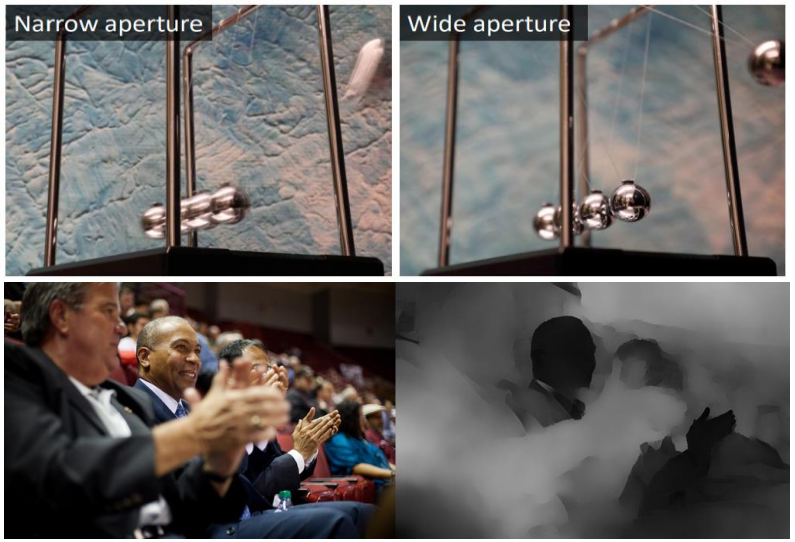
Introduction

- What is image restoration?
  - Corrupted image로부터 high quality clean image를 복원하는 inverse problem (ill-posed)
    - Denoising
    - Deblurring
    - Deraining
  - DL 적용 전/후
    - Regularization, optimization problem
      - Prior 사용 등 hand-craft하게 적용되는 부분들 존재
    - Dataset 및 network에 따라 성능이 결정
      - Task에 따라 dataset에 국한되는 성능을 보임
      - MPRNet, HINet 등 image restoration task를 동시에 수행하는 네트워크



Introduction

- What is image deblurring?
  - Types of blur
    - Defocus blur
    - Motion blur
      - Uniform/non-uniform blur
        - ✓Levin / Lai dataset
      - Real-world synthesized blur
        - ✓GoPro / RealBlur / HIDE dataset



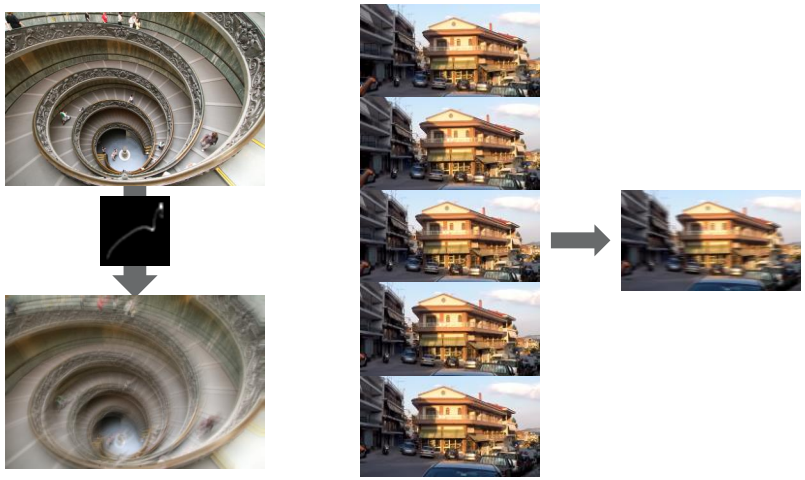
< defocus delbur에 대한 예시 >

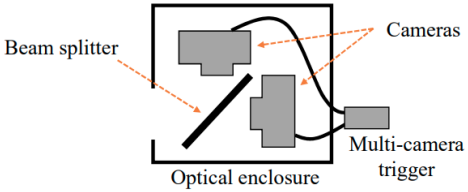
< Lai dataset blur/sharp image pair >

< GoPro dataset blur/sharp image pair >

Introduction

- What is image deblurring?
  - Conventional dataset
    - Levin dataset / Lai dataset
    - Sharp image로부터 blur kernel을 convolution하여 이미지 획득
  - Real-world synthesized dataset
    - GoPro dataset / REDS dataset / RealBlur dataset
    - 취득한 sharp image의 연속적 프레임들의 평균을 통해 blurry image 생성





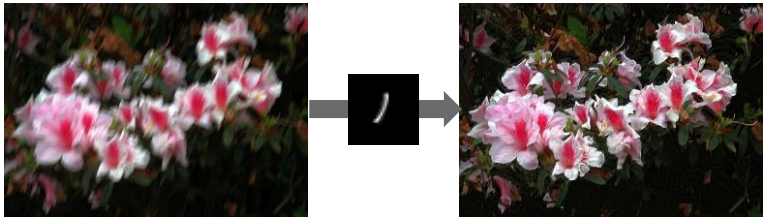
(a) A diagram of our image acquisition system

< blur/sharp dataset 생성 과정 (Lai / GoPro / ReaBlur dataset) >

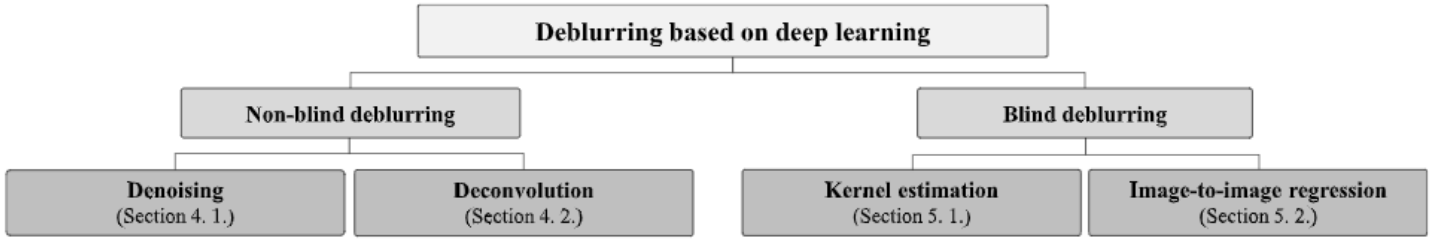
Introduction

- What is image deblurring?
  - Methods of deblurring
    - Non-blind deblurring
      - Denoising + deconvolution
      - Blur kernel을 알고 있다는 가정 하에 deconvolution 및 denoising에 초점을 둠
    - Blind deblurring
      - Kernel estimation + non-blind deblurring
      - Image-to-image regression
      - Blur kernel을 모른다고 가정, kernel estimation 단계 추가

< Levin dataset blur kernel 및 GT image >



< Blur kernel estimation + non-blind deblurring >



< Deblurring algorithms classified into four categories [1] >

Background

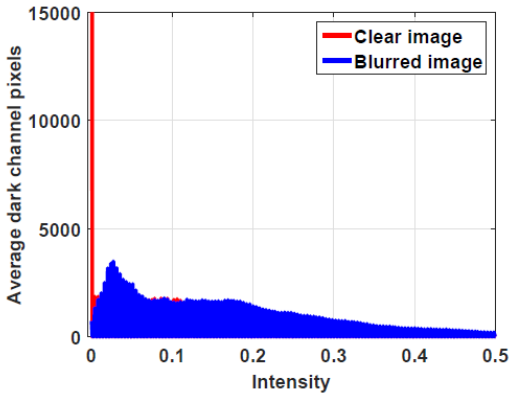
- Conventional deblurring [2, 3]
  - Motion blur modeling
    - Blurry image B는 latent image L과 blur kernel K의 convolution과 noise N으로 modeling

$$B = I \otimes k + n$$

  - Blurred / sharp image prior 사용
    - Blurred / Sharp image의 prior, 즉 이미지 특성을 반영
    - '사람이 관찰을 통해 얻어낸 사전 지식'

(a) (b) (c)

(a) Clear (b) Blurred



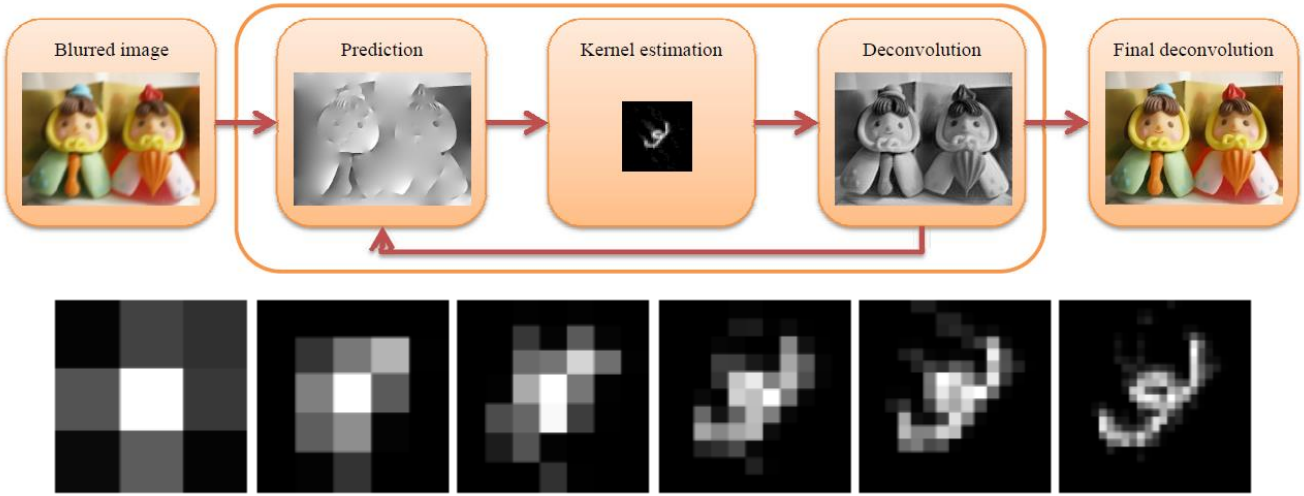
< Using dark channel prior (dehazing 및 deblurring)>

Background

- Conventional deblurring [2, 3]
  - Regularization term을 추가한 optimization problem 풀기 (MAP 및 FFT)

$$\min_{I,k} \|I \otimes k - B\|_2^2 + \gamma\|k\|_2^2 + \mu\|\nabla I\|_0 + \lambda\|D(I)\|_0$$

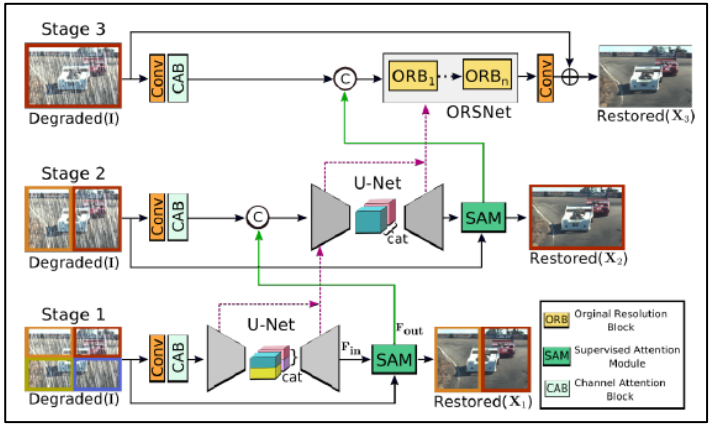
  - Kernel 및 restored image update
    - Coarse-to-fine method로 kernel 및 image를 iterative하게 update

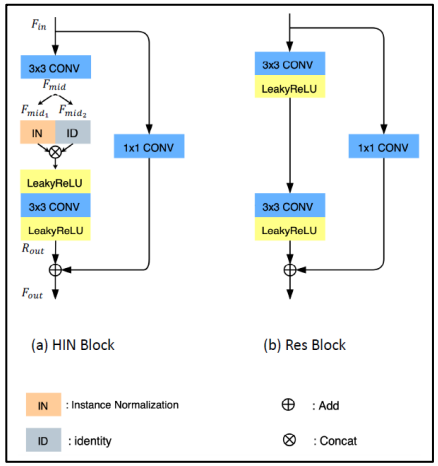


< Overall deblurring process & kernel estimation >

Background

- End-to-end DL method [4, 5]
  - MPRNet [4]
    - Image를 patch별로 잘라서 cross-stage간 feature fusion 및 supervised attention module 적용
  - HINet [5]
    - Mini-batch 안의 image patch간 variance가 크기 때문에 사용되지 않는 batch normalization 대신 instance normalization 채택
  - GoPro dataset / HIDE dataset 등 특정 dataset에서 학습되어서 다른 형태의 blurry image에 대해 성능 하락을 보임





< MPRNet network architecture, HINet의 HIN Block idea>

Exploring Blur Kernel Space

- Modeling blur kernel
  - MAP based models
    - In-wild blurry image 및 blur 강도가 큰 이미지를 복원하는데 한계점 존재
      - Optimization problem을 푸는 과정에서 x와 k는 여러 답안을 낼 수 있는 ill-posed problem
      - Convolution operator로만 blur kernel을 정의

$$B = I \otimes k + n$$

  - Deep-learning models
    - Cross-domain tasks에 부적합
      - REDS dataset에 학습된 네트워크는 GoPro dataset에 대해 좋지 않은 성능을 보임
      - Kernel overfitting 발생
  - Blur operator family
    - Blur kernel generator도 학습하자!

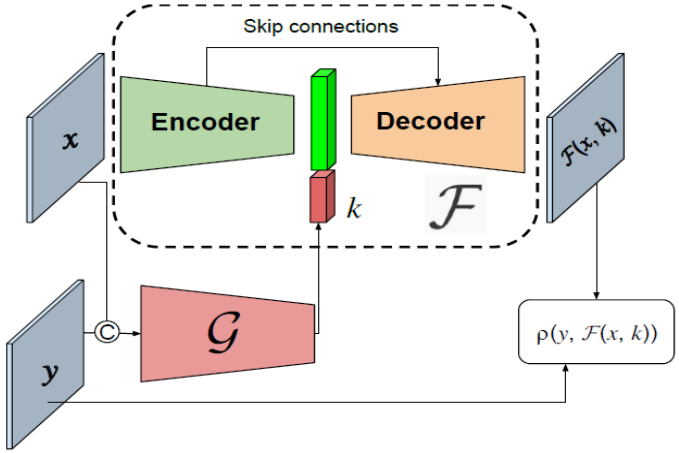
Exploring Blur Kernel Space

- Learning the blur operator family
  - Blur operator family $\mathcal{F}$ + Kernel extractor $\mathcal{G}$
    - $\mathcal{F}$와 $\mathcal{G}$를 sharp / blurry input image (x_i, y_i)로 동시에 학습

$$y_i = \mathcal{F}(x_i, k_i), \quad k_i = \mathcal{G}(x_i, y_i)$$

    - Blurry image y_i와 생성한 "fake" blurry image $\mathcal{F}(x_i, \mathcal{G}(x_i, y_i)$의 차이를 최소화하는 방향

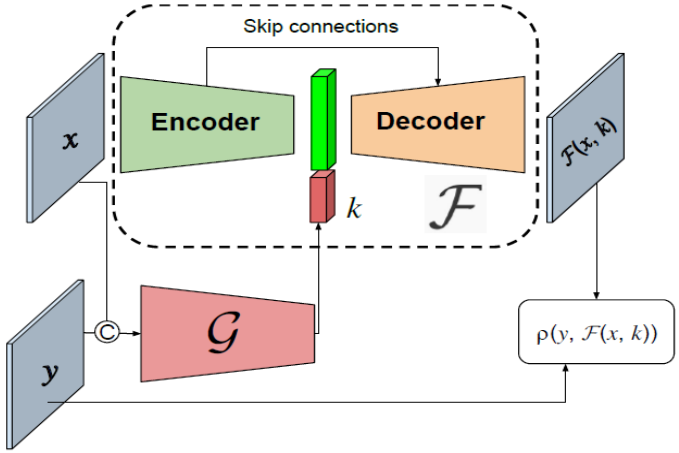
$$Minimize \sum_{i=1}^{n} \rho(y_i, \mathcal{F}(x_i, \mathcal{G}(x_i, y_i))$$



< Blur operator $\mathcal{F}$ & blur kernel extractor $\mathcal{G}$ >

Exploring Blur Kernel Space

- Learning the blur operator family
  - Blur operator family $\mathcal{F}$ + Kernel extractor $\mathcal{G}$
    1. Concatenation of (x, y) pair $\mathcal{G}$ 통과하여 encoded kernel vector k 생성
    2. $\mathcal{F}$의 encoder를 통해 bottleneck embedding vector 생성
    3. k와 concatenation 후 decoder를 통해 fake blurry image $\mathcal{F}(\mathrm{x}, \mathrm{k})$ 생성
    4. Loss 최소화: $\sum_{i=1}^{n} \rho(y_i, \mathcal{F}(x_i, \mathcal{G}(x_i, y_i))$



< Blur operator $\mathcal{F}$ & blur kernel extractor $\mathcal{G}$ >

Exploring Blur Kernel Space

- Blind image deblurring
  - Blur operator family $\mathcal{F}$ 및 Kernel extractor $\mathcal{G}$는 고정
  - Kernel k와 sharp image x update
    - 고정된 z_x, z_k로부터 $G^x_{\theta_x}, G^k_{\theta_k}$ update하여 x, k update
    - $x = G^x_{\theta_x}(z_x)$
    - $k = G^k_{\theta_k}(z_k)$
  - GAN-inversion image restoration
    - $G^k_{\theta_k}$ 로 StyleGAN pretrained model 사용

Algorithm 1 Blind image deblurring

Input: blurry image y

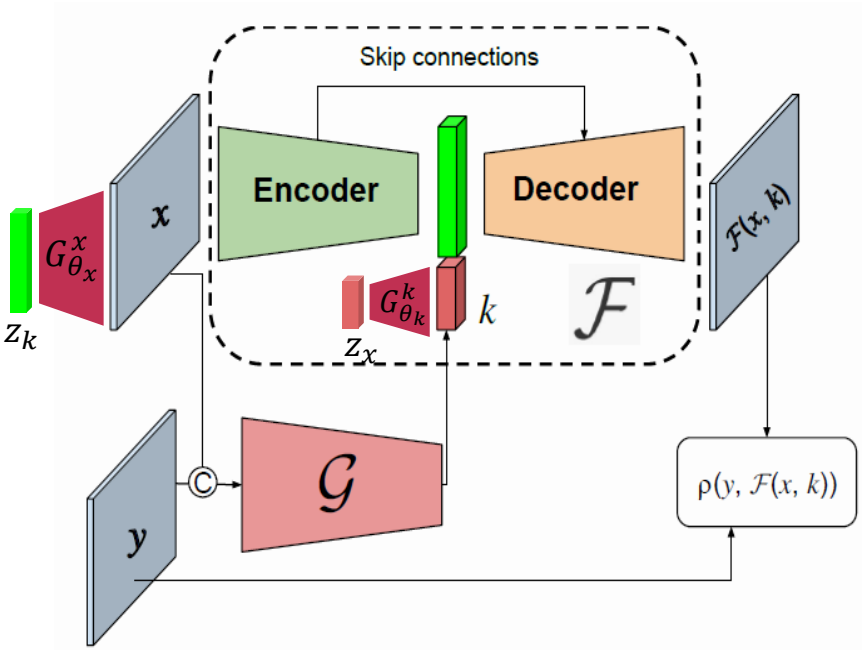
Output: sharp image x

Sample $z_x \sim \mathcal{N}(0, I)$
Randomly initialize θ_x of $G^x_{\theta_x}$
while θ_x has not converged **do**
    Sample $z_k \sim \mathcal{N}(0, I)$
    Randomly initialize θ_k of $G^k_{\theta_k}$
    while θ_k has not converged **do**
        $g_k \leftarrow \partial\mathcal{L}(\theta_x, \theta_k)/\partial\theta_k$
        $\theta_k \leftarrow \theta_k + \alpha * ADAM(\theta_k, g_k)$
    end while
    $g_x \leftarrow \partial\mathcal{L}(\theta_x, \theta_k)/\partial\theta_x$
    $\theta_x \leftarrow \theta_x + \alpha * ADAM(\theta_x, g_x)$
end while
$x = G_{\theta_x}(z_x)$

< Blind image deblurring algorithm >

Exploring Blur Kernel Space

- Blind image deblurring



< Blur operator $\mathcal{F}$ & blur kernel extractor $\mathcal{G}$ >

Algorithm 1 Blind image deblurring

Input: blurry image y

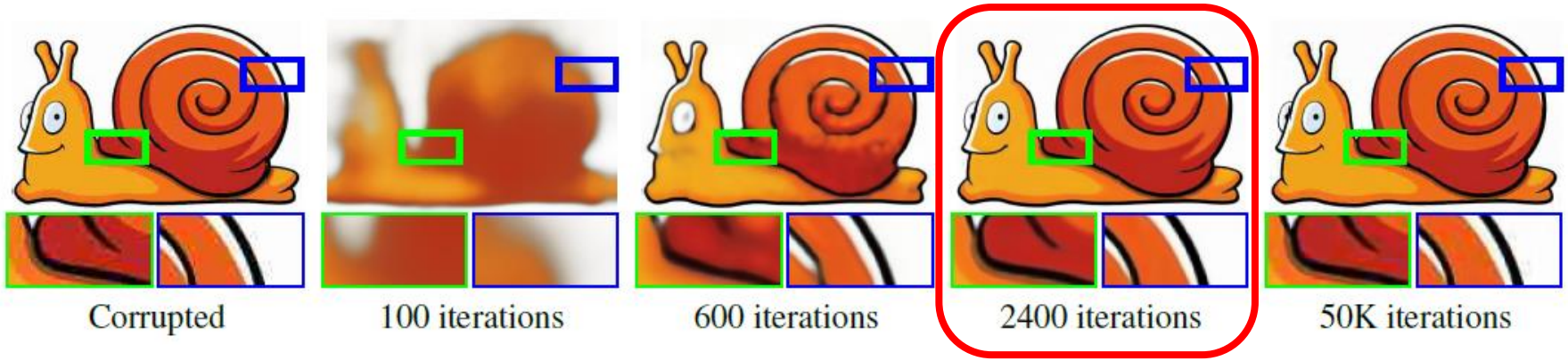
Output: sharp image x

Sample $z_x \sim \mathcal{N}(0, I)$
Randomly initialize θ_x of $G^x_{\theta_x}$
while θ_x has not converged **do**
  Sample $z_k \sim \mathcal{N}(0, I)$
  Randomly initialize θ_k of $G^k_{\theta_k}$
  while θ_k has not converged **do**
    $g_k \leftarrow \partial\mathcal{L}(\theta_x, \theta_k)/\partial\theta_k$
    $\theta_k \leftarrow \theta_k + \alpha * ADAM(\theta_k, g_k)$
  end while
  $g_x \leftarrow \partial\mathcal{L}(\theta_x, \theta_k)/\partial\theta_x$
  $\theta_x \leftarrow \theta_x + \alpha * ADAM(\theta_x, g_x)$
end while
$x = G_{\theta_x}(z_x)$

< Blind image deblurring algorithm >

Exploring Blur Kernel Space

- Deep Image Prior [6]



- $Original\ Loss:\ \rho(y_i, \mathcal{F}(x_i, \mathcal{G}(x_i, y_i)$
  - Regularization term 추가
  - 최종 $Loss: \rho(y_i, \mathcal{F}(x_i, \mathcal{G}(x_i, y_i) + \lambda\|k\|_2 + \gamma(g_u^2(x) + g_v^2(x))^{\alpha/2}$
    - Optimization 안정화 및 trivial solution을 피하기 위한 L2 norm of k
    - Hyper-Laplacian prior of image gradients of x
    - γ, λ, α: hyperparameters

Experiments

- Retrieving unseen kernel
  - Blur operator에 대한 신뢰성 검사
    - SRN-Deblur [7]에 대해 blur-swap된 dataset으로 training 진행 후 성능 확인
  - Cross domain kernel extraction
    - 다른 dataset에 대해 blur operator family 적용
      - Reconstructed blurry image와 original blur image와의 PSNR 비교

< GoPro dataset 및 REDS dataset에서 학습된 blur operator을 사용해 얻은 retrieved blur >

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

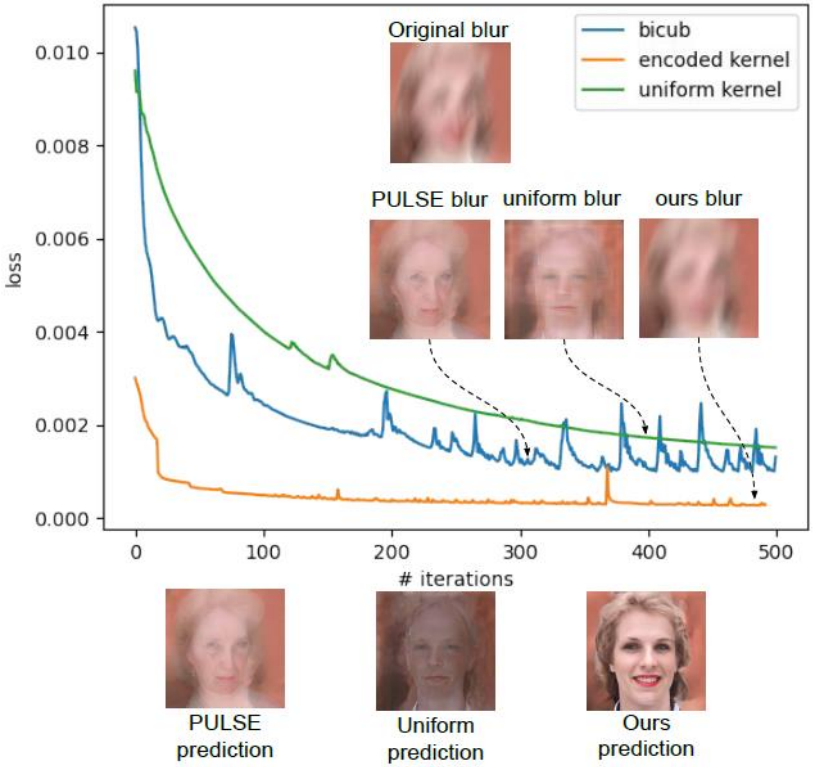
Table 2. Results of SRN-Deblur trained [36] on the original and blur-swapped datasets.

	Test set	
Tranining set	REDS4	GOPRO
REDS	34.35	30.67
GOPRO	31.38	35.13

Table 3. Results of our method in retrieving unseen blur kernel with same and cross-domain configs.

Experiments

- Loss convergence
  - GAN manifold를 사용해서 성능이 잘 나온 것이지 않나?
    - Yes
    - Blur kernel prior 고려하는 것 또한 중요
      - Uniform / bicubic / encoded kernel
      - 빠르게 converge 되고 deblur 결과도 잘 나옴
    - Real world blur
      - Linear X
      - Uniform X
  - 논문 대부분의 그림들이 face image에 대해 진행
    - Pretrained StyleGAN manifold를 사용하지 않은 경우 성능이 비교적 안 좋음



Experiments

- Qualitative results

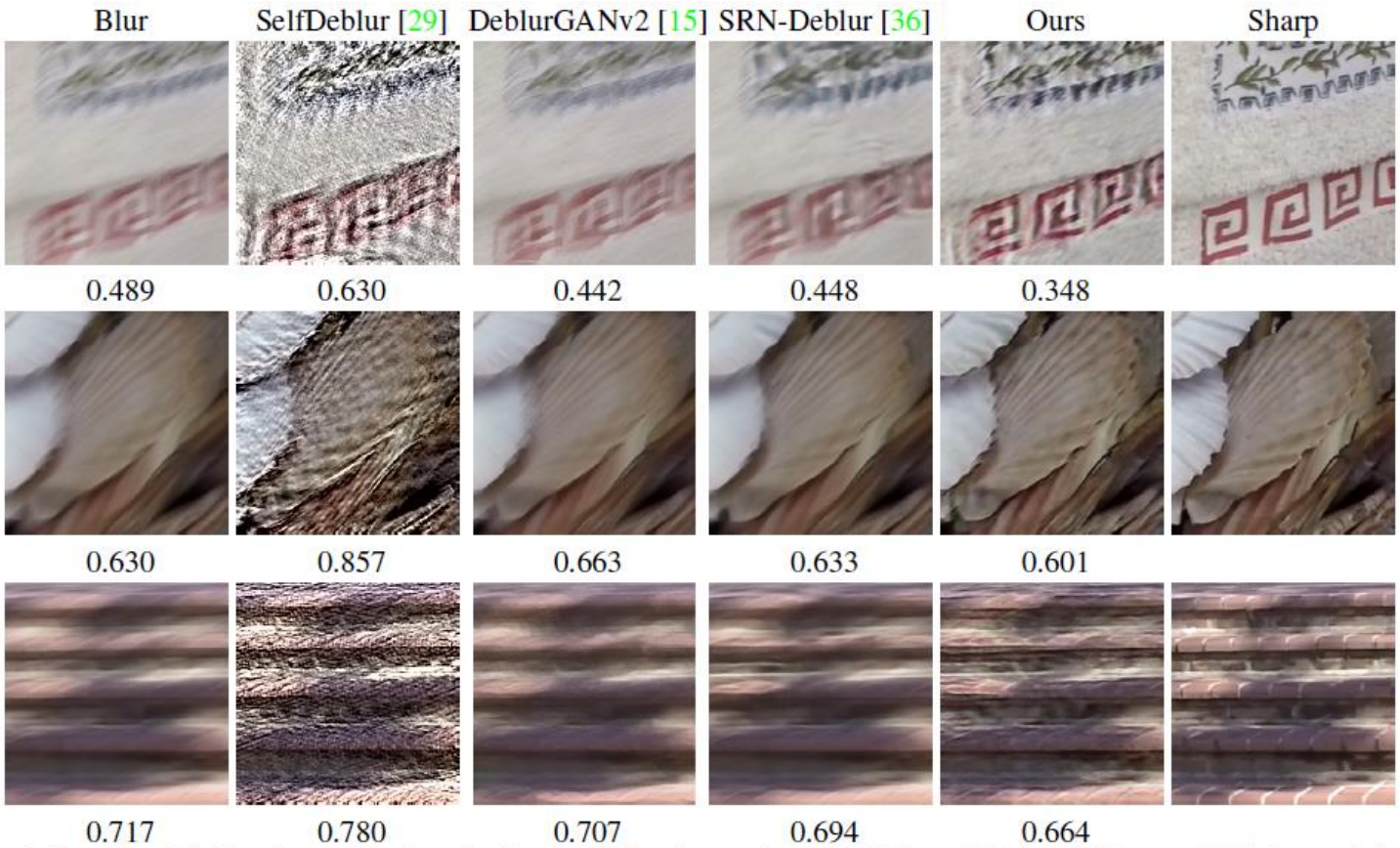
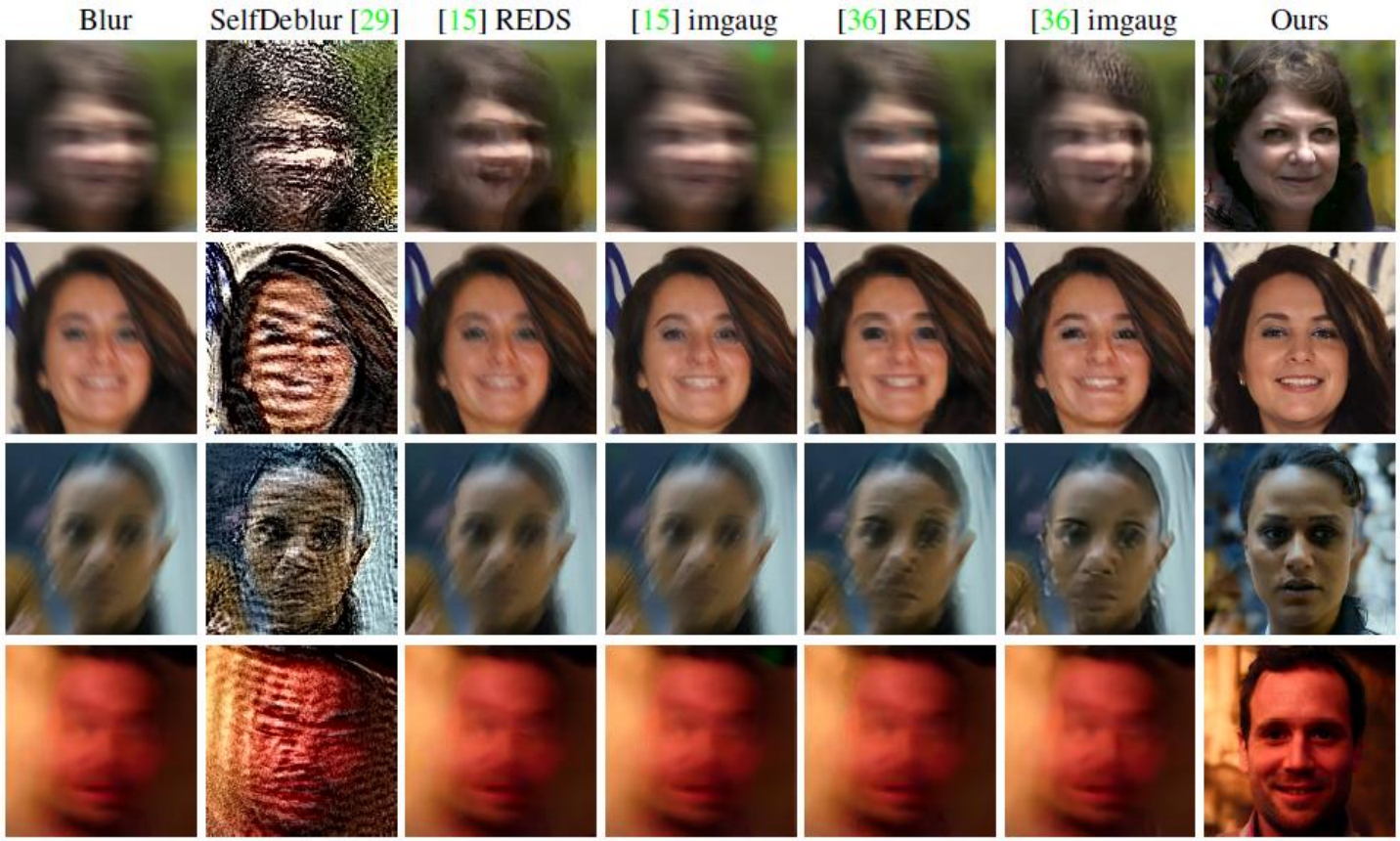


Figure 3. Results of deblurring methods trained on REDS and tested on GOPRO, and their LPIPS score [42] (lower is better).

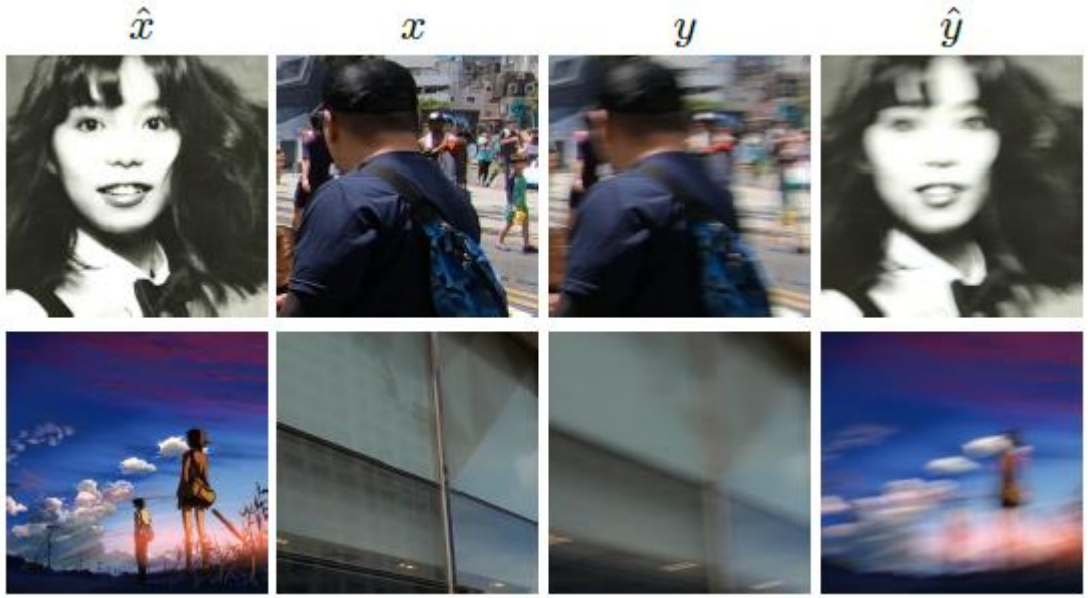
Experiments

- Qualitative results



Experiments

- Blur synthesis
  - Transferring blur kernel
    - Source image pair (x, y) → target image pair $(\hat{x}, \hat{y})$ 이미지 생성
      - Real world blur transfer가 가능하다는 점
      - Data augmentation으로 활용 가능



Exploring Blur Kernel Space

- Network Architecture & training details
  - Blur operator $\mathcal{F}$ & blur kernel extractor $\mathcal{G}$
    - Training: 6×10^6 iterations / Nvidia V100 5GB로 4일
    - Inference: Nvidia V100 210s
  - 128 배수 image resolution input

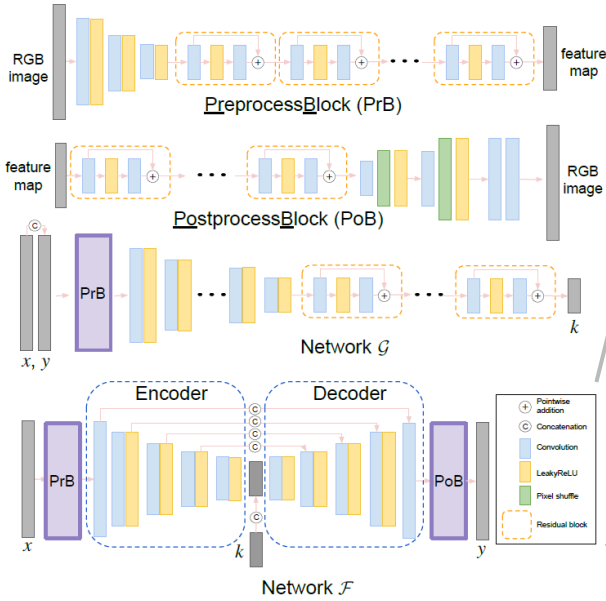


Figure 2. Detailed architecture of the proposed method

Encoder	
Layer	Output shape
PreprocessBlock	$64 \times H/4 \times W/4$
Conv(64, 64, 3, 2, 1)	$64 \times H/8 \times W/8$
LeakyReLU(0.1)	$64 \times H/8 \times W/8$
Conv(64, 128, 3, 2, 1)	$128 \times H/16 \times W/16$
LeakyReLU(0.1)	$128 \times H/16 \times W/16$
Conv(128, 256, 3, 2, 1)	$256 \times H/32 \times W/32$
LeakyReLU(0.1)	$256 \times H/32 \times W/32$
Conv(256, 512, 3, 2, 1)	$512 \times H/64 \times W/64$
LeakyReLU(0.1)	$512 \times H/64 \times W/64$
Conv(512, 512, 3, 2, 1)	$512 \times H/128 \times W/128$
LeakyReLU(0.1)	$512 \times H/128 \times W/128$
Decoder	
Layer	Output shape
TransConv(1024, 512, 3, 2, 1)	$512 \times H/64 \times W/64$
LeakyReLU(0.1)	$512 \times H/64 \times W/64$
TransConv(1024, 256, 3, 2, 1)	$256 \times H/32 \times W/32$
LeakyReLU(0.1)	$256 \times H/32 \times W/32$
TransConv(512, 128, 3, 2, 1)	$128 \times H/16 \times W/16$
LeakyReLU(0.1)	$128 \times H/16 \times W/16$
TransConv(256, 64, 3, 2, 1)	$64 \times H/8 \times W/8$
LeakyReLU(0.1)	$64 \times H/8 \times W/8$
TransConv(128, 64, 3, 2, 1)	$64 \times H/4 \times W/4$
LeakyReLU(0.1)	$64 \times H/4 \times W/4$
PostprocessBlock	$64 \times H \times W$

Table 4. Structure of the encoder and decoder of $\mathcal{F}$

Conclusion

- Encoding blur kernel space
  - Arbitrary dataset에 대해서 encoded space를 활용해 deblurring을 진행할 수 있음
  - Linear, uniform하지 않은 real world blur를 다룰 수 있는 장점
- Blur synthesis
  - Transferring blur kernel
  - Data augmentation
- 한계점
  - Pretrained StyleGAN manifold를 사용하기 때문에 face image에 대해 한정적으로 성능이 잘 나오는 점
    - Prior 및 optimization, regularization 관점으로는 기존 방법을 그대로 적용
  - 128 배수의 resolution을 가진 이미지를 input으로 넣어줘야 함
    - GoPro dataset과 같이 128의 배수가 아닌 이미지에 대해 진행 방법에 대해 구체적 기술 X
  - 후속 논문들 및 더 좋은 성능의 논문들이 있지만 SRN-deblur(2018) 논문과 비교한 점

References

[1] Koh, Jaihyun, Jangho Lee, and Sungroh Yoon. "Single-image deblurring with neural networks: A comparative survey." *Computer Vision and Image Understanding* 203 (2021): 103134.

[2] Cho, Sunghyun, and Seungyong Lee. "Fast motion deblurring." *ACM SIGGRAPH* Asia 2009 papers. 2009. 1-8.

[3] Pan, Jinshan, et al. "Blind image deblurring using dark channel prior." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[4] Verma, Monu, Santosh Kumar Vipparthi, and Girdhari Singh. "Hinet: Hybrid inherited feature learning network for facial expression recognition." *IEEE Letters of the Computer Society* 2.4 (2019): 36-39.

[5] Zamir, Syed Waqas, et al. "Multi-stage progressive image restoration." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

[6] Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

[7] Tao, Xin, et al. "Scale-recurrent network for deep image deblurring." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

Thank you