SAMSUNG ADVANCED INSTITUTE OF TECHNOLOGY

KAIST

ICLR

Controllable Blur Data Augmentation Using 3D-Aware Motion Estimation

Insoo Kim [1,2], Hana Lee [1], Hyong-Euk Lee [1], Jinwoo Shin [2]

Insoo1.kim@samsung.com

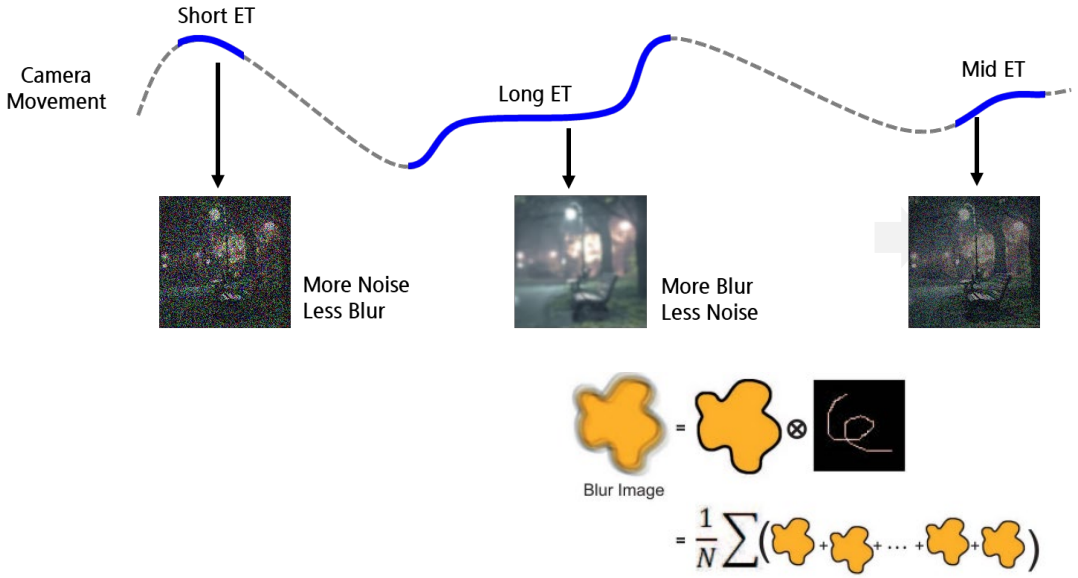
[1]Samsung Advanced Institute of Technology (SAIT)

[2]Korea Advanced Institute of Science and Technology (KAIST)

Introduction : Motion Blur

■ Motion Blur

Mainly caused by the camera shake and object movement in the long exposure time



Blind Image Deblurring Problem

Solve $y = x \otimes k$ with two unknowns, e.g., sharp image x and blur kernel k

- Basically, it is an ill-posed problem

Introduction : Blur Datasets

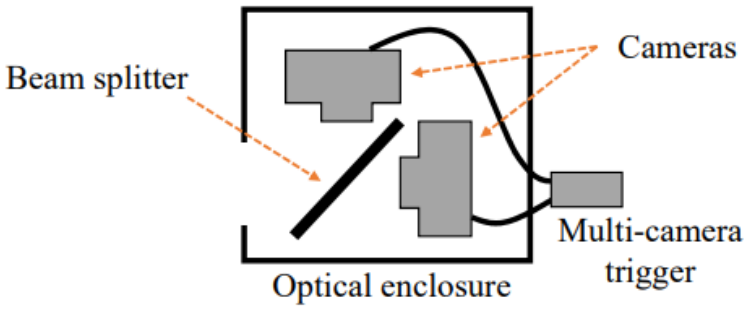
■ The Essence of Large-Scale Blur Datasets for Practical Usage

1) Under the low-light conditions, it is challenging to simultaneously capture the same scene for blur and sharp images

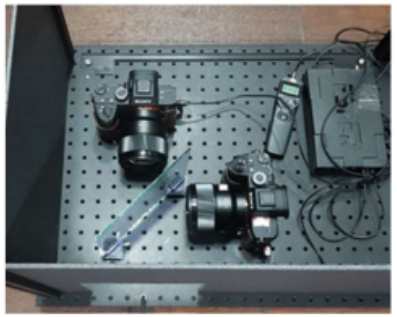
2) To overcome this, the dual camera system [1] was introduced by using the beam splitter

- Photometric & geometric alignment (i.e., post-processing) between blur and sharp images is necessary

- This heavy system restricts from collecting large-scale dataset → insufficient number of scenes & blur patterns



(a) A diagram of our image acquisition system



(b) Our image acquisition system

3) Alternatively, we can consider the data augmentation scheme like high-level vision tasks

- There have been little studies particularly focused on blur data augmentation

[1] Rim et al., “Real-World Blur Dataset for Learning and Benchmarking Deblurring Algorithms” (ECCV 20)

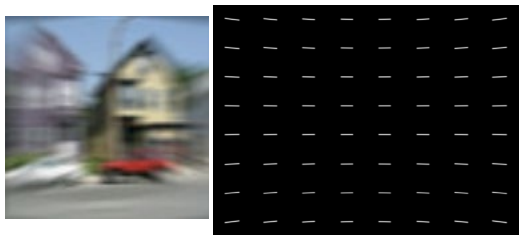
Introduction : Blur Modeling

■ Blur Modeling for Data Augmentation

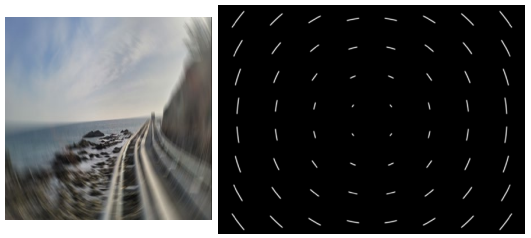
Camera Motion (3D Rotation / Translation)

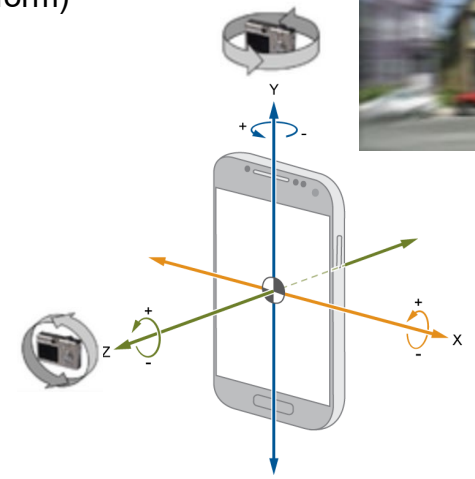
3D motion (uniform) → 2D projective transform (non-uniform)

Out-of-plane rotation (y-axis rotation)

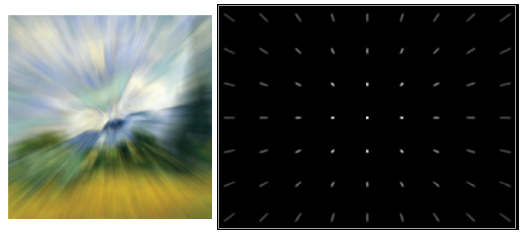


In-plane rotation (z-axis rotation)





Forward motion (z-axis translation)



Motivation: camera and object motion inherently arise in 3D space

For example, in the case of camera motion,

- **Previous methods – 2D non-uniform field → regressing per-pixel kernel (huge ill-posed)**
- **Proposed method – 3D uniform motion trajectory → estimating rotation & translation parameters (much easier)**

Related Works : Kernel-Based Methods

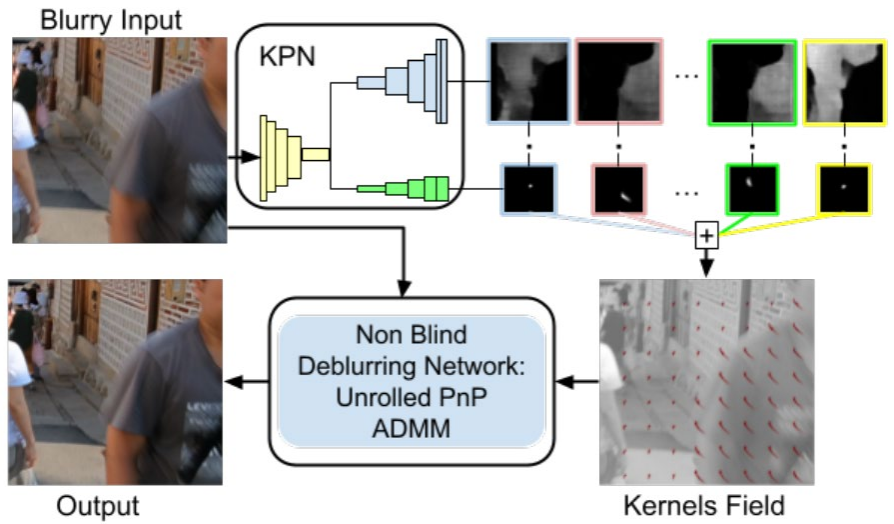
■ Kernel-Based Methods

We may utilize byproduct of the kernel-based methods, i.e., blur kernels, to synthesize blur images

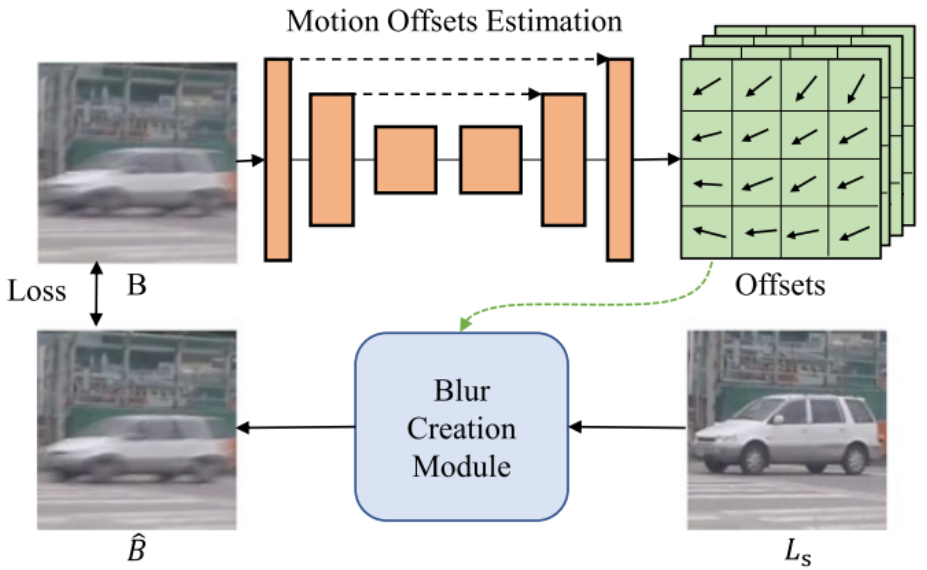
(+) Estimate blur kernels to synthesize blur images ☺

(-) Simply regressing the simulated kernels may not hold in generating realistic blur images ☹

(-) The kernel-based methods are designed to estimate motions from a 2D perspective ☹



J-MKPD [1]



MotionETR [2]

[1] Carbajal et al., "Blind motion deblurring with pixel-wise kernel estimation via kernel prediction networks" (IEEE Trans. on Computational Imaging 2023)
[2] Zhang et al., "Exposure trajectory recovery from motion blur" (TPAMI 2021)

Related Works : Diffusion-Based Blur Synthesis

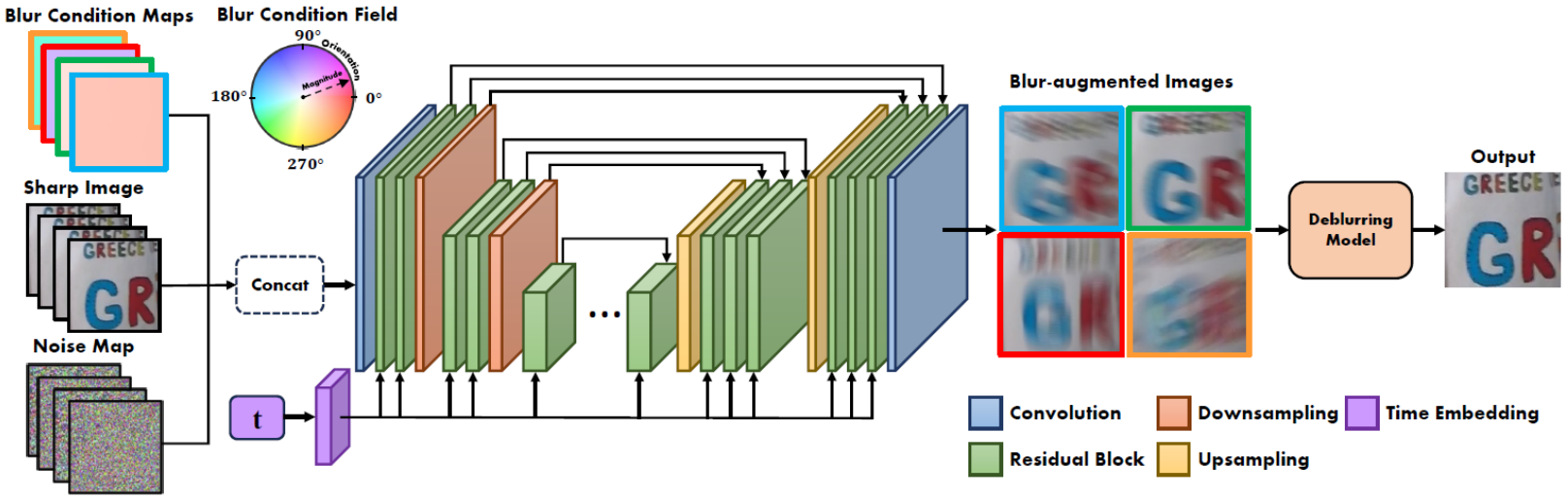
■ Diffusion-Based Blur Data Augmentation

(+) Introducing controllable blur synthesizer, allowing for generating a wide range of unseen blur scenarios

(-) Diffusion-based method is computationally expensive, and challenging to use for real-time blur data augmenta

(-) This scheme requires video frame images which are not typically provided in blur datasets

(-) 3D aspects of blur modeling are not considered



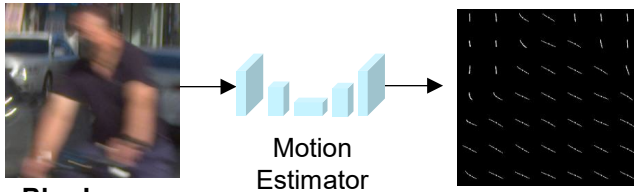
ID-Blau [1]

[1] Wu et al., “Id-blau: Image deblurring by implicit diffusion-based reblurring augmentation” (CVPR 2024)

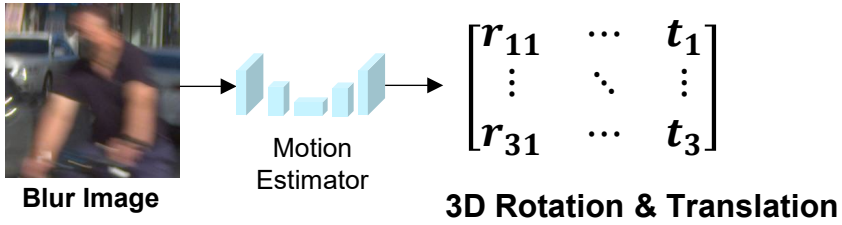
Proposed Method : Main Idea & Key Challenges

Main Idea

Estimating complex 2D per-pixel blur kernels 



Estimating simple 3D camera positions 



Key Challenges

Challenge I : motion estimation in 3D space

We only have 2D blur images and no 3D information → How do we estimate 3D motion using blur images only?

Challenge II : controllability

Controllability is essential for data augmentation → How to effectively control motions?

Proposed Method: Overview

■ Controllable 3D-Aware Blur Synthesizer

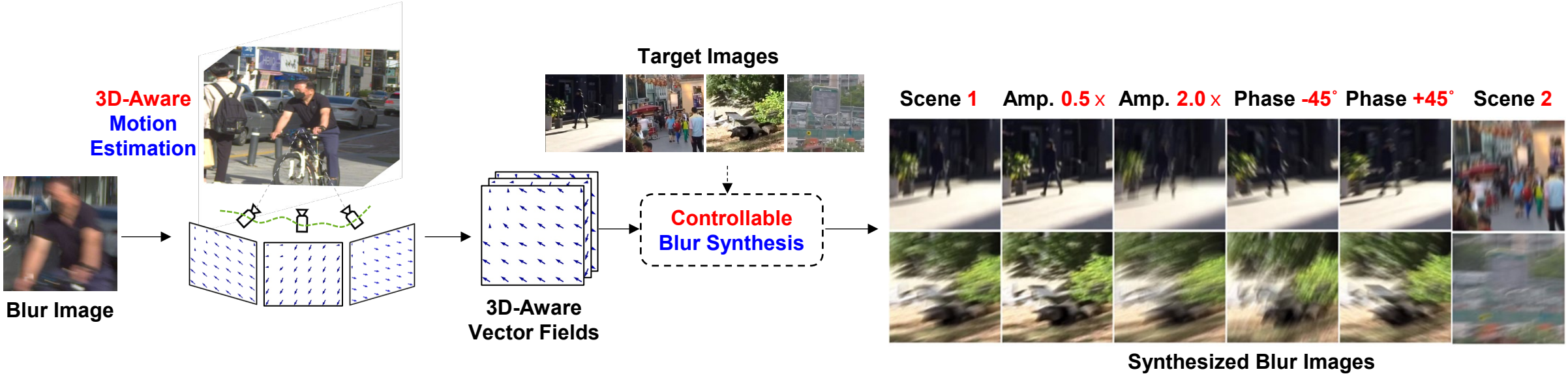
To effectively estimate 3D motion, we decompose 3D motion into 2D motion and 3D residual components

1) Challenge I : motion estimation in 3D space

- ▶ Solve this by estimating 3D residual components via neural networks
- ▶ This allows for estimating 3D motion without requiring explicit depth measurements

2) Challenge II : controllability

- ▶ Solve this by representing a motion as a vector field → vector amplitude (blur amount) and phase (blur direction)
- ▶ Allowing for generation of millions of diverse blur patterns by randomly varying blur amount and directions



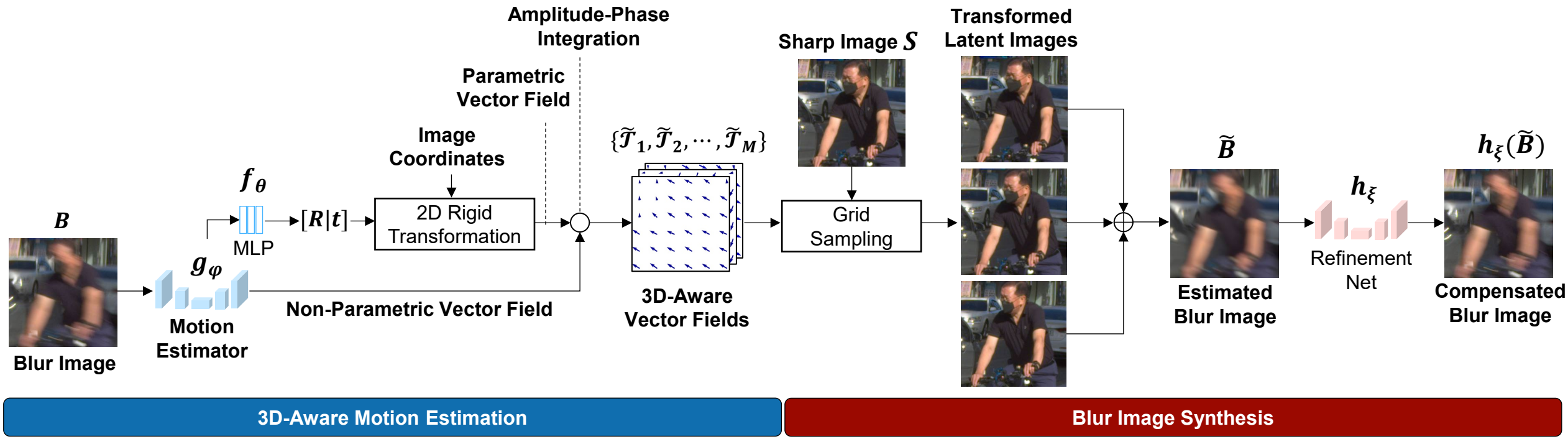
Proposed Method: Overview

■ Overview of Proposed Method

Our blur synthesizer consists of two main components

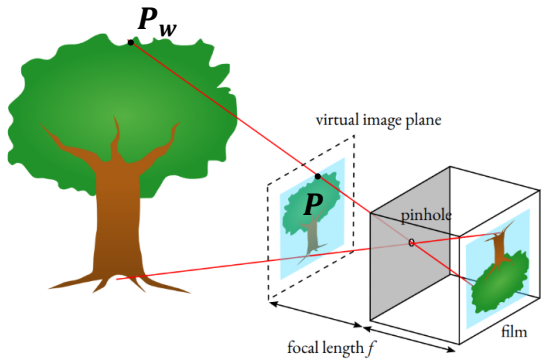
1) 3D-Aware Motion Estimation – 2D parametric vector field + 3D non-parametric vector field

2) Blur Image Synthesis – Corresponding scene image generation and blur image synthesis by aggregating them



Proposed Method: 3D-Aware Blur Model

■ 3D-Aware Motion Estimation (Camera Motion)



World coordinate : X

Image coordinate : u

Camera intrinsics : $\mathbf{K}$

Projection: π

2D rigid transformation parameters

$$\mathcal{T}_\tau(u) = [R_\tau|t_\tau]\mathbf{u} = \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11}x + r_{12}y + t_1 \\ r_{21}x + r_{22}y + t_2 \end{bmatrix}$$

3D rigid transformation = 2D transformation + 3D geometry residual components

$$\mathcal{T}_\tau(X) = [R_\tau|t_\tau]X = \begin{bmatrix} r_{11} & \cdots & t_1 \\ \vdots & \ddots & \vdots \\ r_{31} & \cdots & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11}x + r_{12}y + r_{13}z + t_1 \\ r_{21}x + r_{22}y + r_{23}z + t_2 \\ r_{31}x + r_{32}y + r_{33}z + t_3 \end{bmatrix} = \begin{bmatrix} r_{11}x + r_{12}y + t_1 \\ r_{21}x + r_{22}y + t_2 \\ 0 \end{bmatrix} + \begin{bmatrix} r_{13}z \\ r_{23}z \\ r_{31}x + r_{32}y + r_{33}z + t_3 \end{bmatrix}$$

2D transformation $\mathcal{T}_\tau^*(u)$

3D residual component $\epsilon_\tau(X)$

Since the 3D-aware coordinate vector lies in R^2,

we can simply express it as

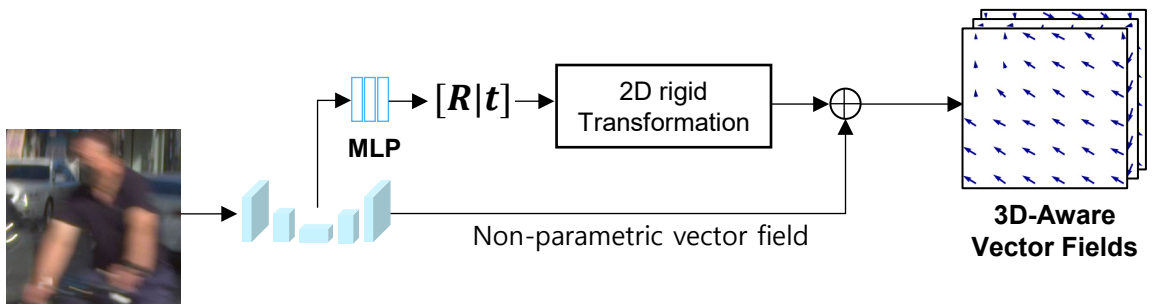
$$\widetilde{\mathcal{T}}_\tau(u) = \pi(\mathcal{T}_\tau^*(u) + \epsilon_\tau(X); K) \approx \mathcal{T}_\tau(u) \circ \epsilon_\tau(u)$$

Neural Network

Amplitude-Phase Integration

3D-aware vector field : $\widetilde{\mathcal{T}}_\tau = \mathcal{T}_\tau \circ \epsilon_\tau$

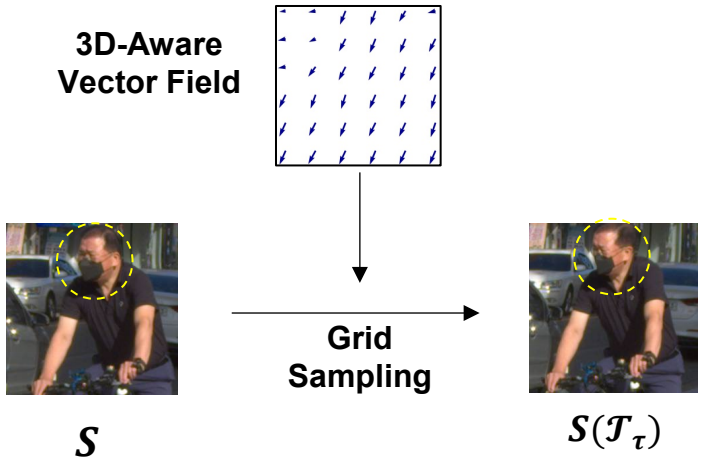
Parametric vector field — **Non-parametric vector field**



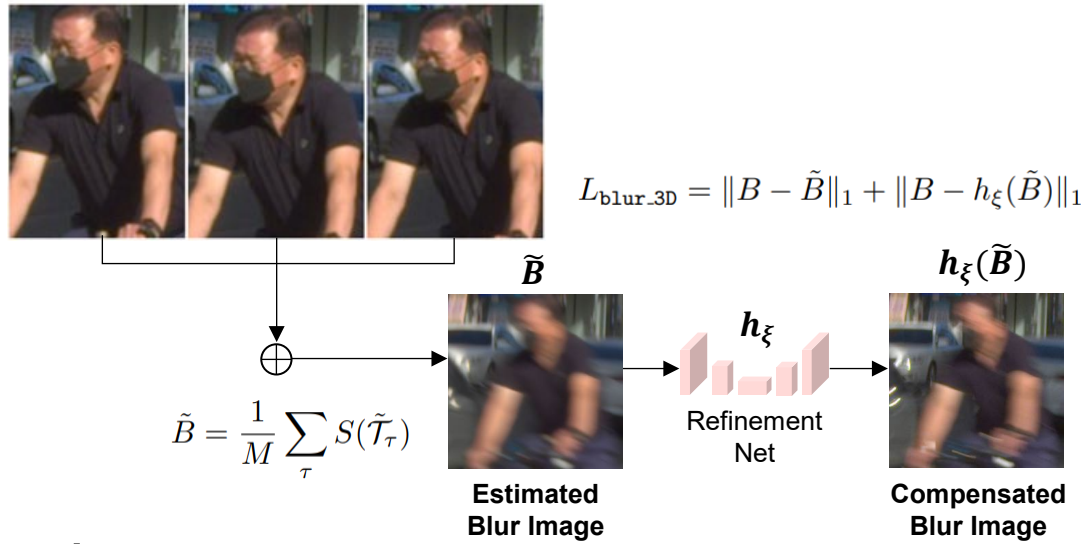
Proposed Method: 3D-Aware Blur Model

■ Blur Image Synthesis (Camera Motion)

Generating the corresponding scene images



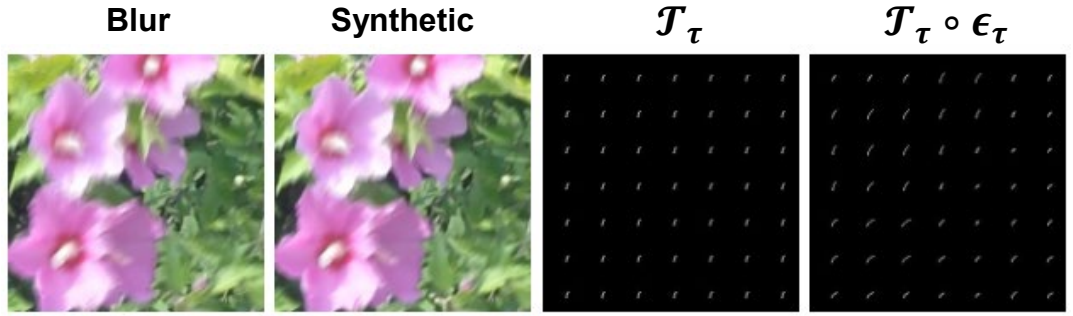
Aggregate the scene images to synthesize a blur image



■ 3D-Aware Motion Estimation (Object Motion)

3D-aware vector field : $\widetilde{\mathcal{T}_\tau} = \mathcal{T}_\tau \circ \epsilon_\tau$

Parametric vector field | **Non-parametric vector field**



▶ **Non-parametric vector field was initially designed for representing 3D geometry residual, but it can handle object motion without further modification**

Proposed Method: Ambiguity Regularization

■ Ambiguity Regularization

The vector field modification introduces ambiguities, resulting in implausible results and artifacts.
These ambiguities lead to unconstrained optimization problems which makes blur synthesis uncontrollable.

3D-aware vector field modification : $\widetilde{\mathcal{T}_\tau} = \mathcal{T}_\tau \circ \epsilon_\tau = \underline{U} + (\Delta\mathcal{T}_\tau \circ \epsilon_\tau) = U + \delta_\tau$

Canonical Coordinates

Displacement Field

[Without regularization]

[With regularization]

Laplacian Regularization

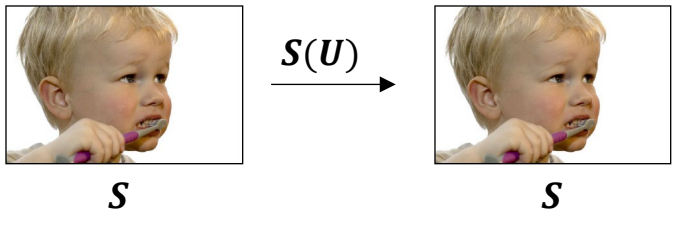
The motion vector field is not irregularly changed in the spatial space

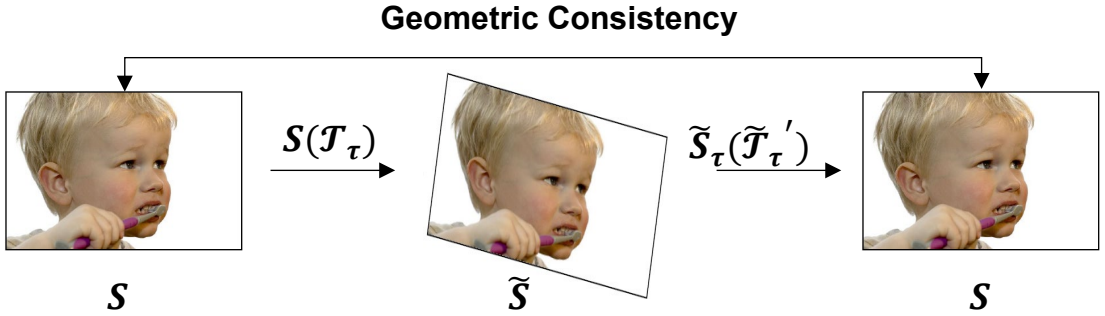
$$L_{\text{smooth}} = \sum_\tau \sum_{x,y} \left(4\tilde{\mathcal{T}}_\tau(x,y) - \sum_{i,j} \tilde{\mathcal{T}}_\tau(x+i,y) + \tilde{\mathcal{T}}_\tau(x,y+j)\right)^2, \qquad i,j \in \{-1,1\}$$

Geometric Regularization

The 3D transformed sharp image should be geometrically reverted to the original sharp image

$L_{\text{geometric}} = \frac{1}{M} \sum_\tau \|S - \tilde{S}_\tau(\tilde{\mathcal{T}}'_\tau)\|_1$, where $\widetilde{\mathcal{T}_\tau}' = U - \delta_\tau$ is the inverse vector field





Proposed Method: Controllable Blur Data Augmentation

■ Controllable Blur Data Augmentation

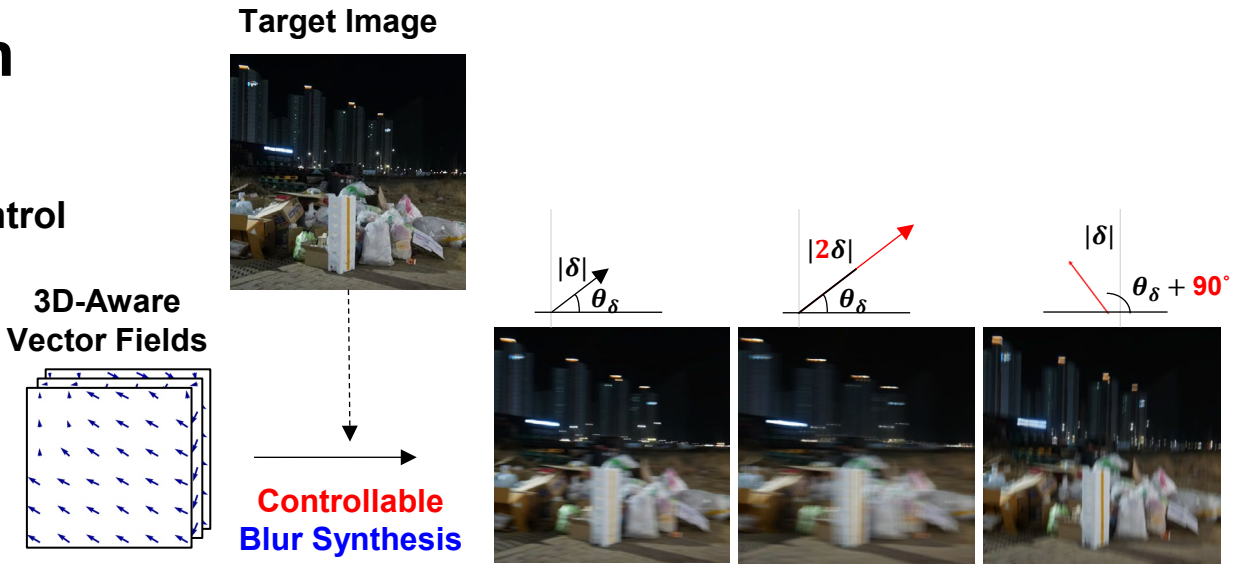
Recall the 3D-aware vector field: $\mathcal{T}_\tau = U + \delta_\tau$

Adjust blur magnitude and direction parameters for blur control

$$\delta = |\delta| \angle \phi(\delta) \longrightarrow \widetilde{\delta}_\tau = (|\delta_\tau| \times \alpha) \angle (\phi(\delta_\tau) + \beta)$$

Finally, we obtain the modified vector field, i.e.,

$$\widetilde{\mathcal{T}}_\tau = U + \widetilde{\delta}_\tau$$



▶ **By randomly varying blur magnitude and directions,**
we can produce a wide range of unseen blur images during the deblurring training

[3D Motion Blur Results]

In-plane rotation blur (z-axis rotation)

[Blur Image] [Synthetic Blur Image] [Blur Trajectory]

Forward motion blur (z-axis translation)

[Blur Image] [Synthetic Blur Image] [Blur Trajectory]

Experimental Results

■ Experimental Results

1) Evaluation Results for Different Datasets and Network Architectures

- ▶ RealBlur : Camera Motion Only
- ▶ GoPro : Camera Motion + Object Motion
- ▶ HIDE : Unseen Motion (Generalization Ability)

Model	GoPro		HIDE		RealBlur-J		RealBlur-R	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
MIMO-UNet+	32.44	0.957	30.00	0.930	31.92	0.919	39.10	0.969
+ GeoSyn (ours)	**33.01**	**0.962**	**30.86**	**0.940**	**32.55**	**0.925**	**39.68**	**0.972**
Restormer	32.92	0.961	31.22	0.942	32.32	0.924	39.47	0.972
+ GeoSyn (ours)	**33.37**	**0.964**	**31.61**	**0.946**	**33.05**	**0.937**	**40.31**	**0.974**
NAFNet	33.69	0.966	31.32	0.943	32.50	0.928	39.89	0.973
+ GeoSyn (ours)	**34.09**	**0.969**	**31.64**	**0.947**	**32.99**	**0.936**	**40.49**	**0.976**
FFTformer	34.21	0.969	31.62	0.946	32.62	0.933	40.11	0.973
+ GeoSyn (ours)	**34.39**	**0.970**	**31.98**	**0.949**	**33.68**	**0.938**	**40.89**	**0.977**

Experimental Results

■ Experimental Results

2) Other dataset (RSBlur)

Methods	GMACs	RSBlur	
		PSNR ↑	SSIM ↑
SRN-Deblur	1434.82	32.53	0.840
MIMO-UNet+	154.41	33.37	0.856
MPRNet	777.01	33.61	0.861
Restormer	141.00	33.69	0.863
Uformer-B	89.50	33.98	0.866
NAFNet-64	63.64	33.97	0.866
+ GeoAug (ours)	63.64	34.23	0.870

3) 2D modeling vs 3D modeling

P – Parametric vector field
NP – Non-parametric vector field

Methods	P	NP	PSNR ↑	SSIM ↑
None			32.35	0.925
2D parametric	✓		32.65	0.932
3D non-parametric		✓	32.61	0.930
3D flat depth	✓		32.66	0.932
3D monocular depth	✓		32.45	0.929
GeoSyn (ours)	✓	✓	**32.97**	**0.936**

4) Comparison with other augmentation methods

Augmentation Types	Datasets	PSNR ↑	SSIM ↑
None	GoPro	33.69	0.966
Adaptive Basis		33.59	0.965
Exposure Trajectory		33.92	0.968
GeoSyn (ours)		**34.08**	**0.969**
None	RealBlur-J	32.35	0.925
Blur Pipeline		32.57	0.931
ID-Blau		32.70	0.932
GeoSyn (ours)		**32.97**	**0.936**

5) Efficient deblurring models

Methods	GMACs	PSNR ↑	SSIM ↑
NAFNet-16	4.0	31.58	0.912
+ GeoSyn (ours)		**31.97**	**0.921**
NAFNet-32	16.0	31.99	0.920
+ GeoSyn (ours)		**32.59**	**0.931**
NAFNet-64	63.5	32.50	0.928
+ GeoSyn (ours)		**32.99**	**0.936**

Experimental Results

- **Qualitative Results on Real-World Blur Images**