Image Deblurring

Seungyong Lee
POSTECH

Contents

- Fast Motion Deblurring (Siggraph Asia 2009)
- Non-uniform Motion Deblurring for Camera Shakes using Image Registration (Siggraph 2011 Talks)
- Text Deblurring (an ongoing project)

Fast Motion Deblurring

Sunghyun Cho
POSTECH

Seungyong Lee
POSTECH

Motion blur

- Camera jitters

Latent sharp image
Blurred image

Image formation model

- Convolution
  - Motion blur kernel
    - Trace of a sensor

$=$

$\otimes$

Blurred image | Latent sharp image | Blur kernel

*$\otimes$: convolution operator

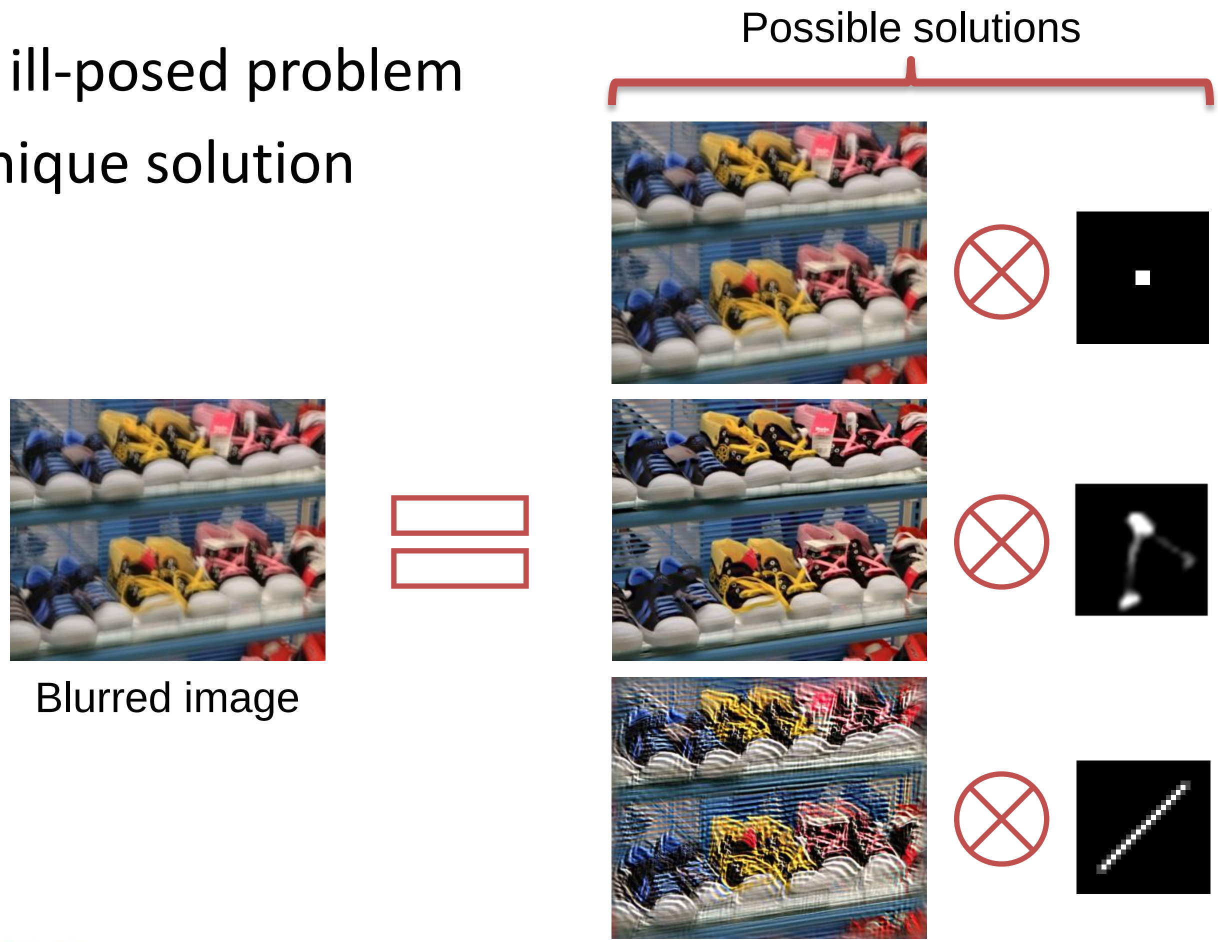
Deblurring

- Non-blind deconvolution
  - Ill-posed (Due to the loss of information caused by motion blur)

- Blind deconvolution
  - **Severely** ill-posed

Blind deconvolution

- Severely ill-posed problem
  - No unique solution



Related work

- Parametric kernels
  - Ex) 1D linear motion blur
  - [Yitzhakey et al. 1998], [Rav-Acha and Peleg 2005], [Cho et al. 2007], [Money and Kang 2008], …

Blurred image = Latent sharp image ⊗ PSF

Related work

- More complex motion blur
  - [Fergus et al. 2006], [Jia 2007], [Shan et al. 2008]
  - Excessive amount of computation

Blurred image　　Latent sharp image　　PSF

Motivation

- Computation time → Important for practical purpose
- Previous methods are slow

Image size: 640 x 480
kernel size: 25 x 25

[Fergus et al. 2006] took 1 hr 25 min.
[Shan et al. 2008] took 4 min 48 sec.

Our method took 5.766 sec. in CPU and 0.734 sec. using GPU accel.

Contributions

- Fast motion deblurring
  - Only a few sec.
  - Fast latent image estimation
  - Fast blur kernel estimation

  → 40x ~ 60x faster than [Shan et al. 2008]

  - GPU acceleration

  → 600x ~ 800x faster than [Shan et al. 2008]

Motion deblurring: Common framework

- Iteratively solve
  1. Estimate a PSF

Blurred image Latent image PSF

  2. Estimate a latent sharp image using a complex image prior

Blurred image Latent image PSF

Motion deblurring: Common framework

- Blur model

$$B = L * K + N$$

Blurred image $\boldsymbol{B}$ Latent image $\boldsymbol{L}$ Blur kernel $\boldsymbol{K}$ Noise $\boldsymbol{N}$

- Energy function

$$f(L,K) = \left|B - L * K\right|^2 + q(L) + r(K)$$

$*$: convolution operator
$q(L)$, $r(K)$: regularization terms or priors for L, K

Motion deblurring: Common framework

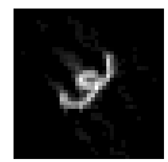
Blurred image

Latent image estimation

$$L' = \arg\min_{L} |B - L * K|^2 + q(L)$$

Kernel estimation

$$K' = \arg\min_{K} |B - L * K|^2 + r(K)$$



Deblurred result

Motion deblurring: Common framework

Blurred image

Kernel estimation

Motion deblurring: Common framework

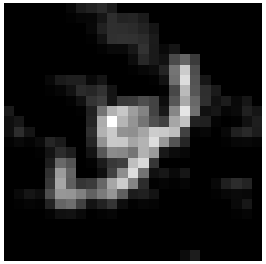
1st latent image estimation

Kernel estimation

Motion deblurring: Common framework

1st latent image estimation

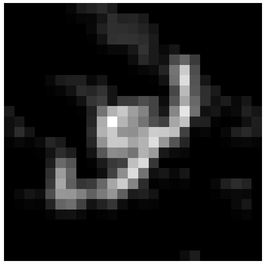
1st kernel estimation



Motion deblurring: Common framework

3rd latent image estimation

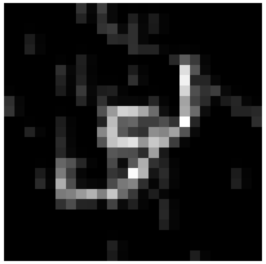
1st kernel estimation



Motion deblurring: Common framework

3rd latent image estimation

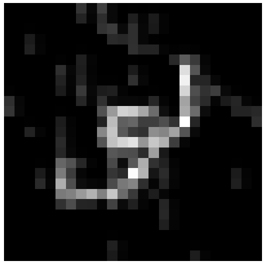
3rd kernel estimation



Motion deblurring: Common framework

5th latent image estimation

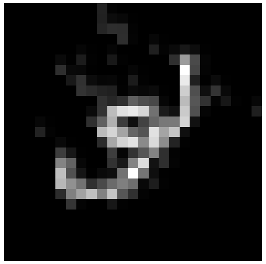
3rd kernel estimation



Motion deblurring: Common framework

5th latent image estimation

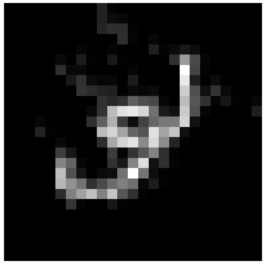
5th kernel estimation



Motion deblurring: Common framework

7th latent image estimation

5th kernel estimation



Motion deblurring: Common framework

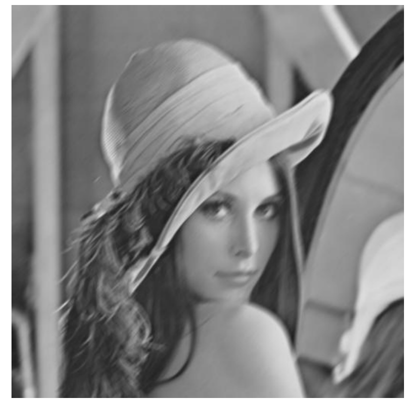
7th latent image estimation

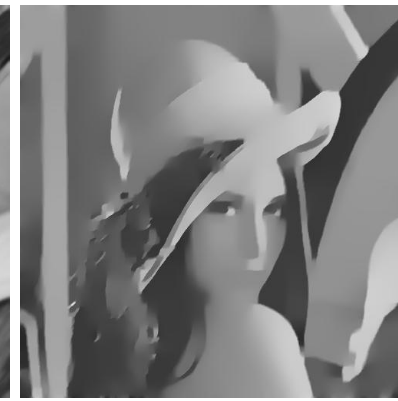
7th kernel estimation

We analyzed and accelerated both estimation steps

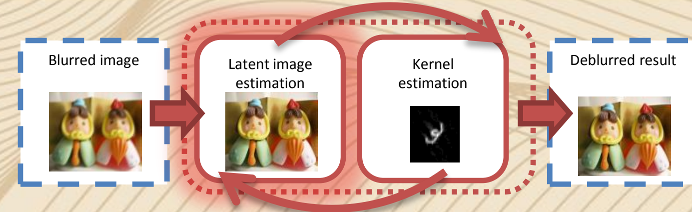
Latent image estimation: Analysis of prev. methods

- Two important properties
  - Restoration of strong edges
    - Inspecting around strong edges, we can find a blur kernel
  - Noise suppression in smooth regions
    - Avoids the effect of noise on kernel estimation

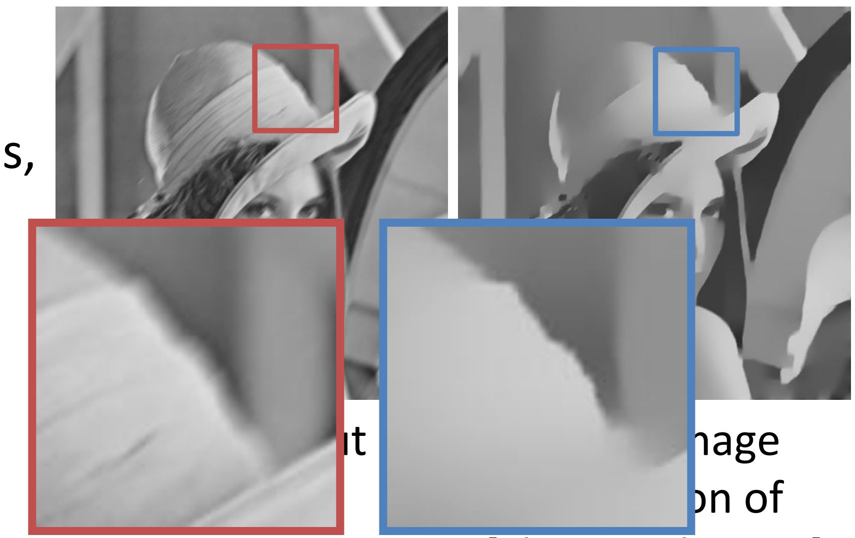

Blurry input


Latent image estimation of [Shan et al. 2008]

Latent image estimation: Analysis of prev. methods



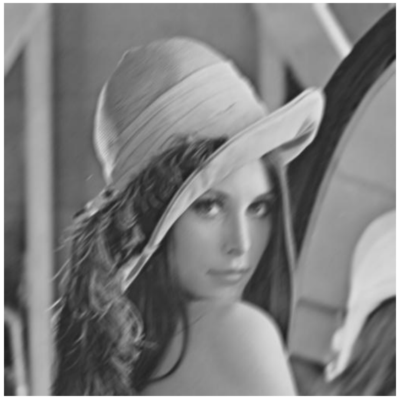
- Two important properties
  - **Restoration of strong edges**
    - Inspecting around strong edges, we can find a blur kernel
  - Noise suppression in smooth regions
    - Avoids the effect of noise on kernel estimation



t
age
on of
[Shan et al. 2008]

Latent image estimation: Analysis of prev. methods

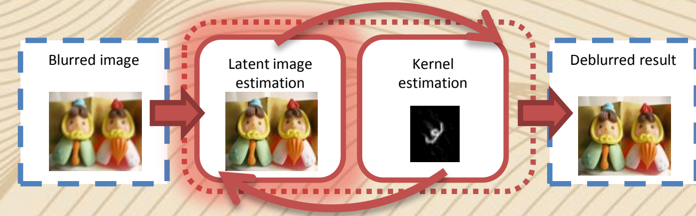
- Two important properties
  - Restoration of strong edges
    - Inspecting around strong edges, we can find a blur kernel
  - **Noise suppression in smooth regions**
    - Avoids the effect of noise on kernel estimation



Blurry input

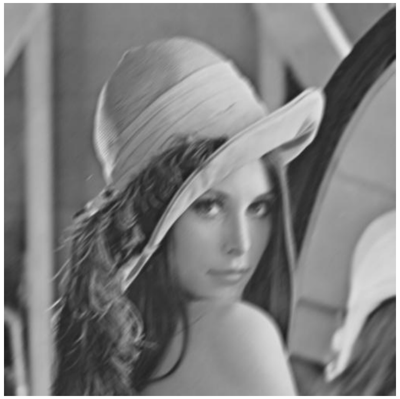
Latent image estimation of [Shan et al. 2008]

Latent image estimation: Analysis of prev. methods



- Two important properties
  - Restoration of strong edges
    - Inspecting around strong edges, we can find a blur kernel
  - Noise suppression in smooth regions
    - Avoids the effect of noise on kernel estimation
- **Computationally expensive priors for** $q(L)$

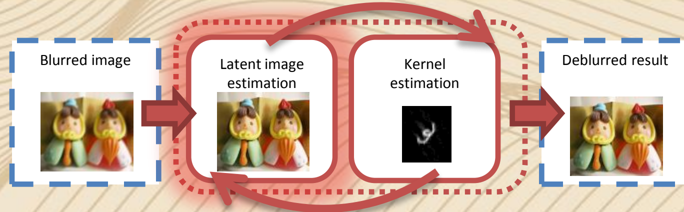
$$\boxed{L' = \arg\min_{L} \left| B - L * K \right|^2 + q(L)}$$



Blurry input

Latent image estimation of [Shan et al. 2008]

Latent image estimation: Basic idea for acceleration



We divide…

Latent image estimation

Simple Deconvolution

Removes blur quickly

Low-quality results

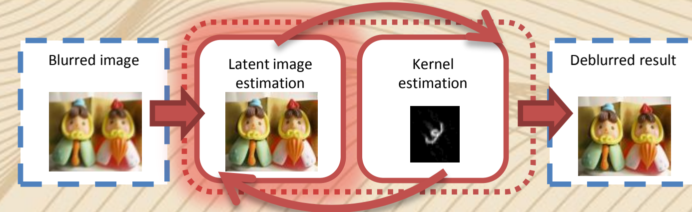
Prediction

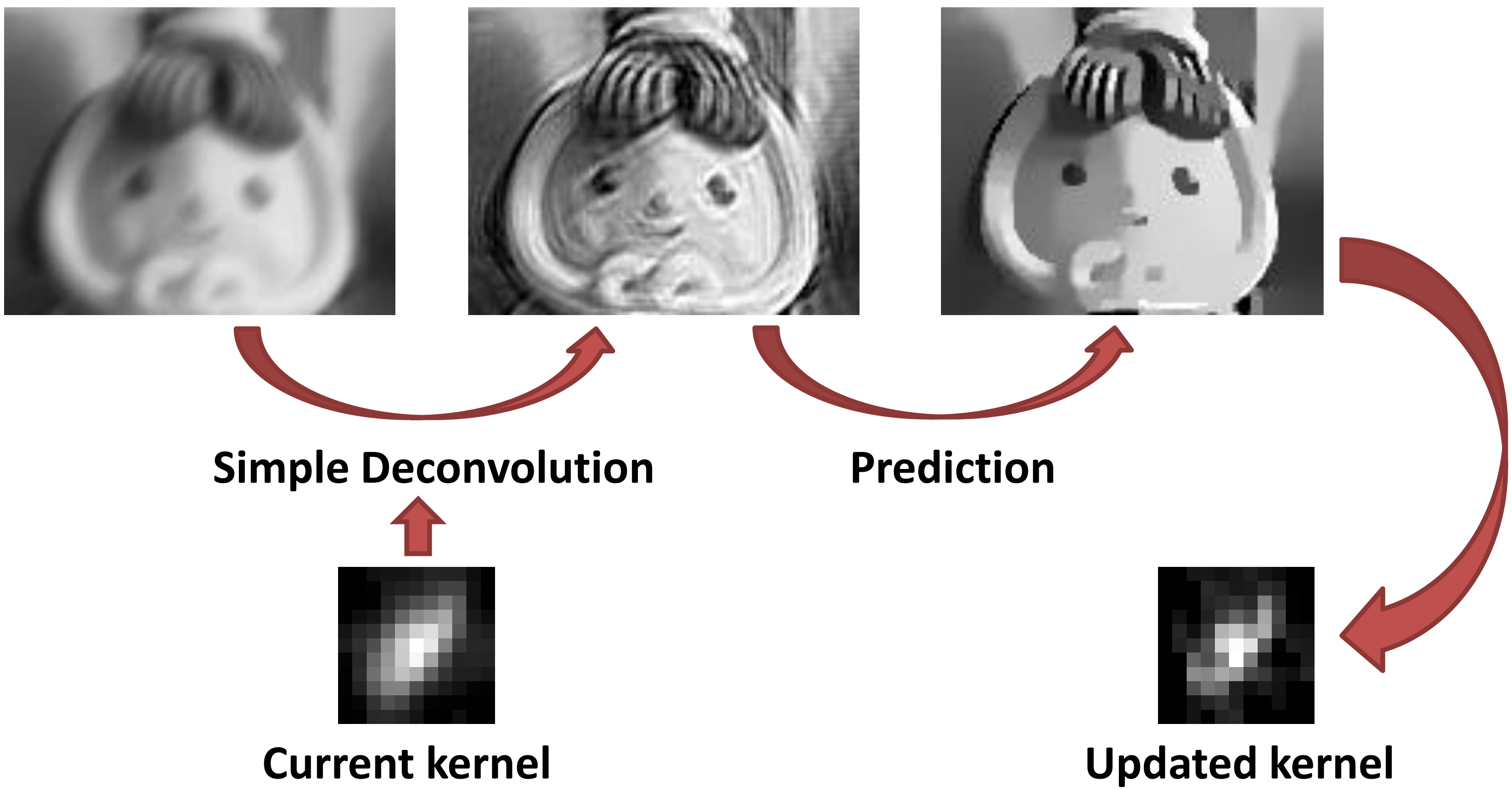
Restores strong edges

Removes noise

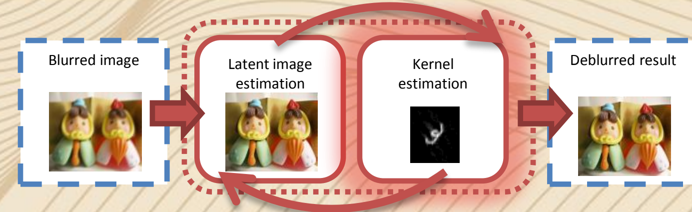
Simple image processing tools

Latent image estimation: Basic idea for acceleration





Kernel estimation: Analysis of prev. methods

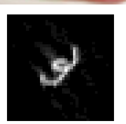


- Previous methods estimate a blur kernel K by optimizing:

$$K' = \arg\min_{K} \left| B - L * K \right|^2 + r(K)$$

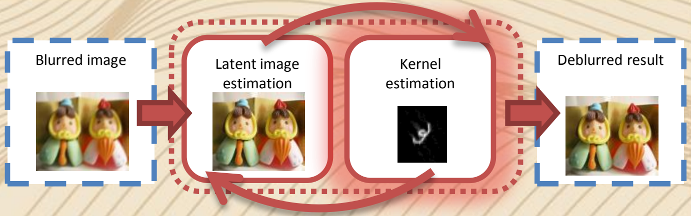
- B : a blurred image

- L : a latent sharp image



- K : a blur kernel
- $r(K)$: a regularization term for K

Kernel estimation: Analysis of prev. methods



- Previous methods estimate a blur kernel K by optimizing:

$$K' = \arg\min_{K} \left| B - L * K \right|^2 + r(K)$$

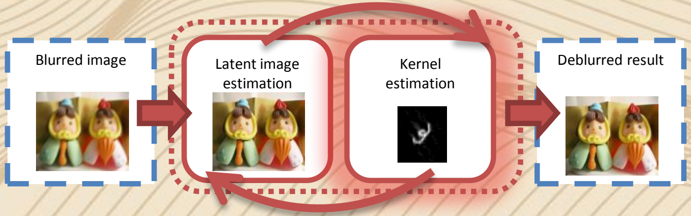
- The simplest case: $r(K) \equiv \alpha |K|^2$ (α: a scalar value)
- Then, K can be found by solving:

$$\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{k} + \alpha\mathbf{k} = \mathbf{L}^{\mathrm{T}}\mathbf{b}$$

$\mathbf{L}$: a matrix rep. of L
$\mathbf{k}$: a vector rep. of K
$\mathbf{b}$: a vector rep. of B

- which can be solved by **a conjugate gradient (CG)** method
- $\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{k}$ is computed **for each CG iteration**

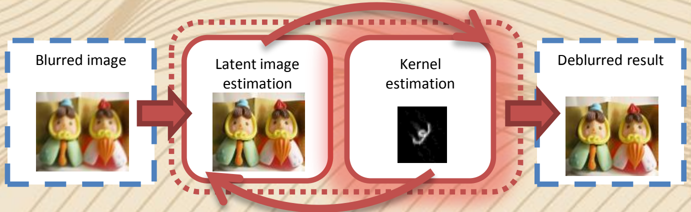
Kernel estimation: Analysis of prev. methods



- Computing $\mathbf{L}^T\mathbf{L}\mathbf{k}$
  - Convolutions & correlations
  - $\mathbf{L}\mathbf{k} \leftarrow L * K$
  - $\mathbf{L}^T\mathbf{L}\mathbf{k} \leftarrow L *_{\text{correl}} (L * K)$
  - Conv. & corr. can be accelerated using FFTs
  - 4 FFTs per CG iter. for computing $\mathbf{L}^T\mathbf{L}\mathbf{k}$

- A CG method needs to iterate…
- **4 FFTs x 30 CG iters = 120 FFTs…**

Kernel estimation: Basic idea for acceleration

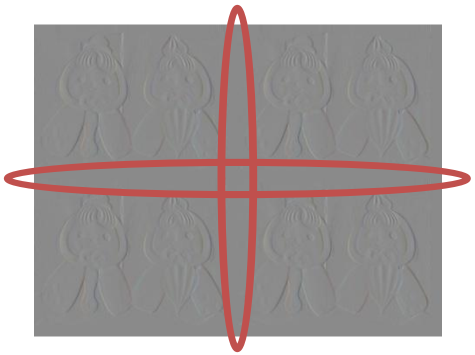


- Energy function using derivative images:

$$K' = \arg\min_{K} \left| \partial B - \partial L * K \right|^2 + a \left| K \right|^2$$

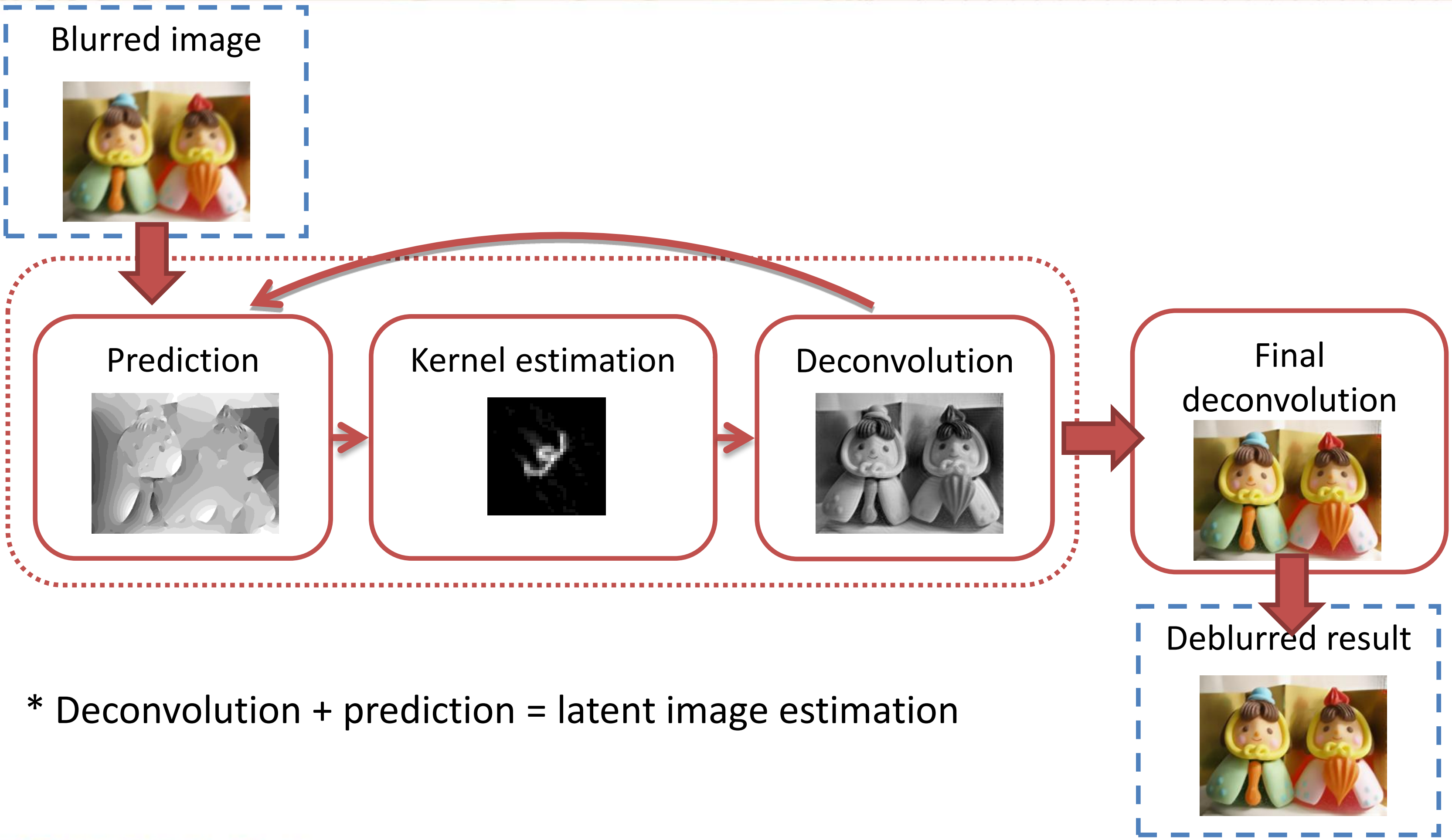
∂: partial differential operator

- With deriv. images, we can avoid boundary problem of FFTs



- $\partial \mathbf{L}^T \partial \mathbf{L} \mathbf{k}$ can be computed using 2 FFTs
- CG iterations converge faster with derivative images

Deblurring process



* Deconvolution + prediction = latent image estimation

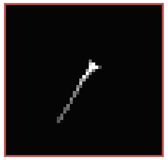
Results

- Implementation
  - CPU version
    - C++, OpenCV, FFTW
  - GPU accelerated version
    - BSGP [Hou et al. 2008] – Easy GPGPU language
    - CUDA FFT library
- Testing environment
  - PC running MS Windows XP 32 bit ver.
  - Intel Core2 Quad CPU 2.66 GHz
  - 3.25GB RAM
  - NVIDIA GeForce GTX 280 Graphics card

Results

Blurry input

Our result



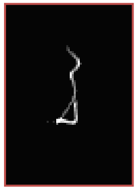
Blur kernel

Image size	Blur kernel size	Processing time (CPU)	Processing time (GPU)
1024 x 768	49 x 47	18.656 sec.	2.125 sec.

Results

Blurry input

Our result



Blur kernel

Image size	Blur kernel size	Processing time (CPU)	Processing time (GPU)
972 x 966	65 x 93	18.813 sec.	5.766 sec.

Results

Blurry input

Our result

Blur kernel

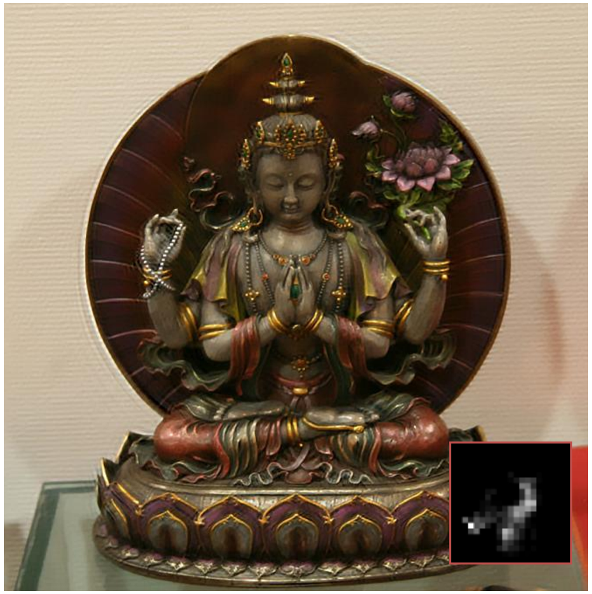
Image size	Blur kernel size	Processing time (CPU)	Processing time (GPU)
858 x 558	61 x 43	8.969 sec.	0.703 sec.

Comparison (quality)

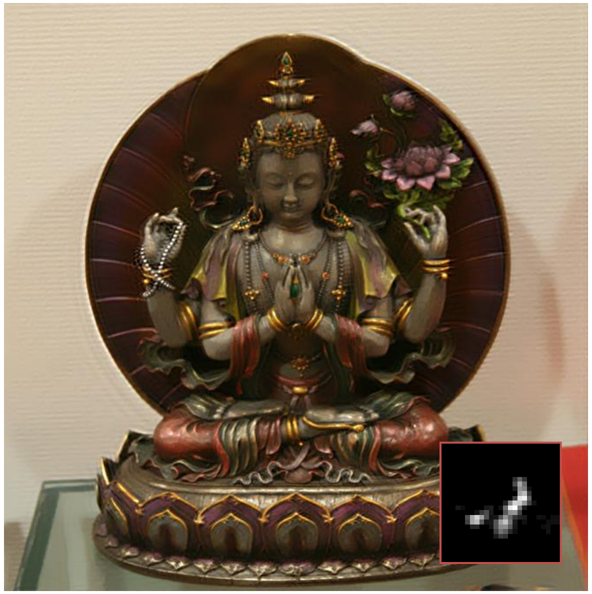
Blurry input

[Yuan et al. 2007]

* This method uses two input images.



[Shan et al. 2008]



our method

Comparison (processing time)

- [Shan et al. 2008] vs. Our method

Image	Size		Processing time (sec.)		
	Image	PSF	Shan et al. (CPU)	Our method (CPU)	Our method (GPU)
Picasso	800 x 532	27 x 19	360	7.485	0.609
Statue	903 x 910	25 x 25	762	15.891	0.984
Night	836 x 804	27 x 21	762	13.813	0.937
Red tree	454 x 588	27 x 27	309	4.703	0.438

* Processing times of our CPU code are updated from our paper

Demo video

- Demo video

Conclusion and future work

- Deblurring method fast enough for practical use
  - Efficient latent image estimation
  - Efficient kernel estimation

- Limitation and future work
  - Sharp edge assumption
  - Noise and saturation
  - Non-uniform blur

Non-uniform Motion Deblurring for Camera Shakes using Image Registration

Sunghyun Cho[1] Hojin Cho[1] Wu-Ying Tai[2] Seungyong Lee[1]

[1]POSTECH [2]KAIST

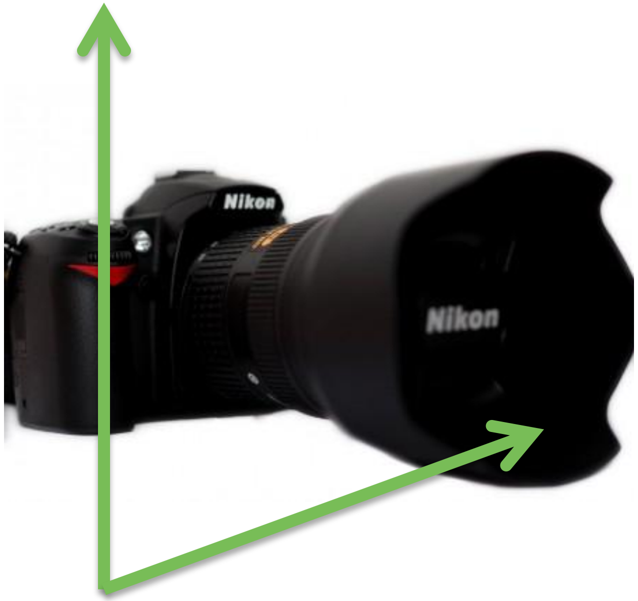
SIGGRAPH2011

Camera Shakes

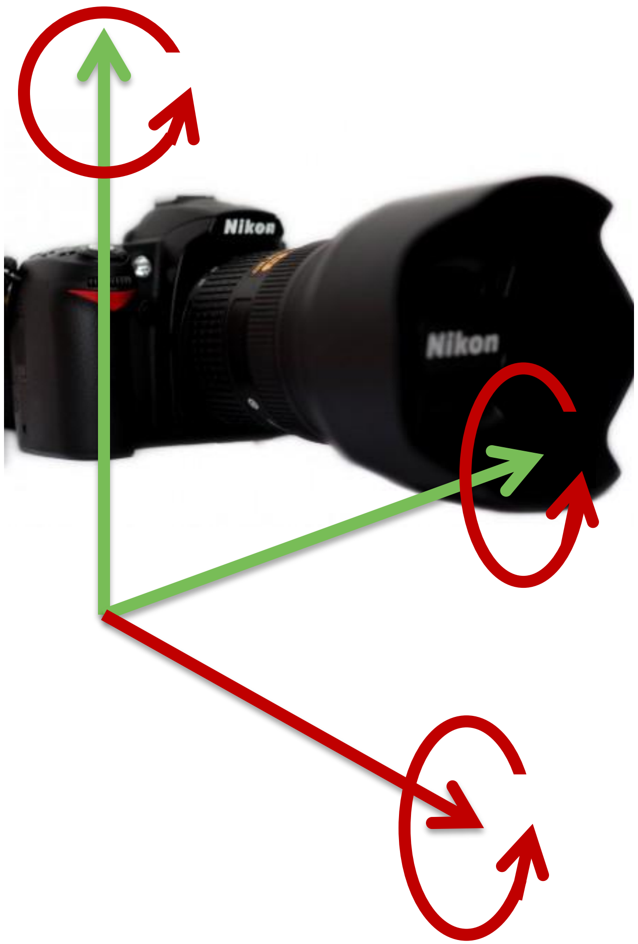
Previous Methods Fail

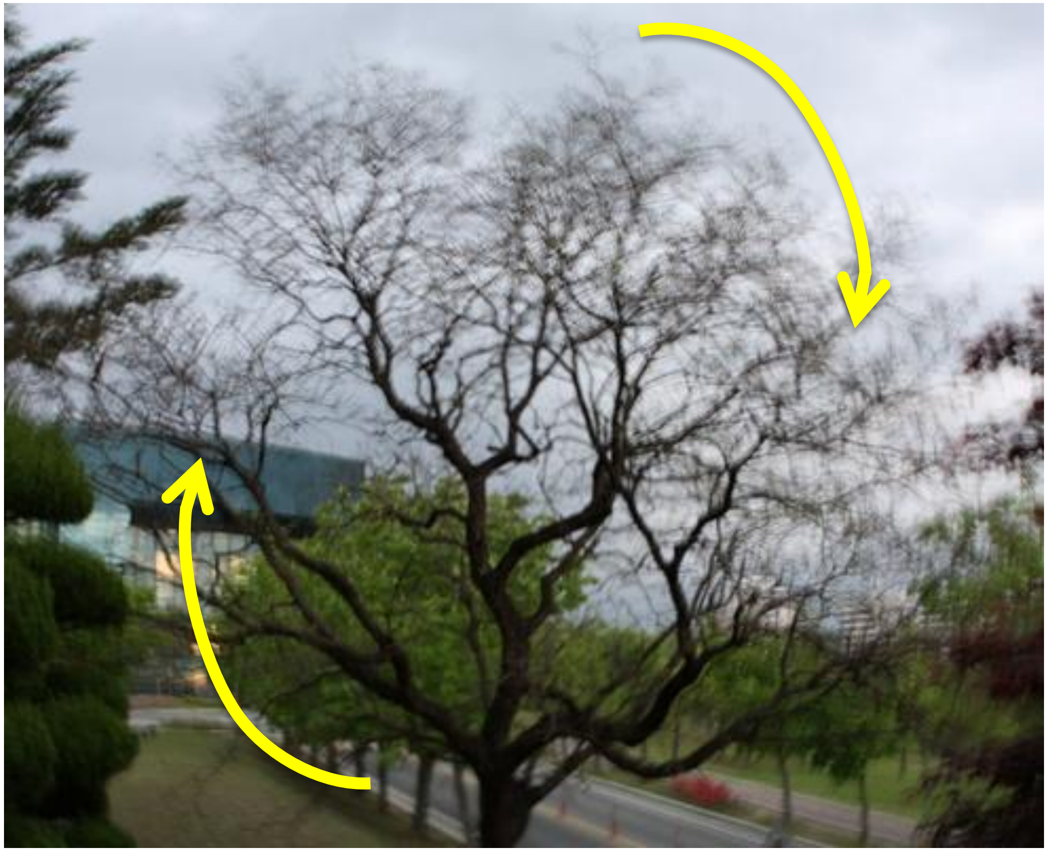
- Result of [Shan et al. 2008]

Uniform Blur

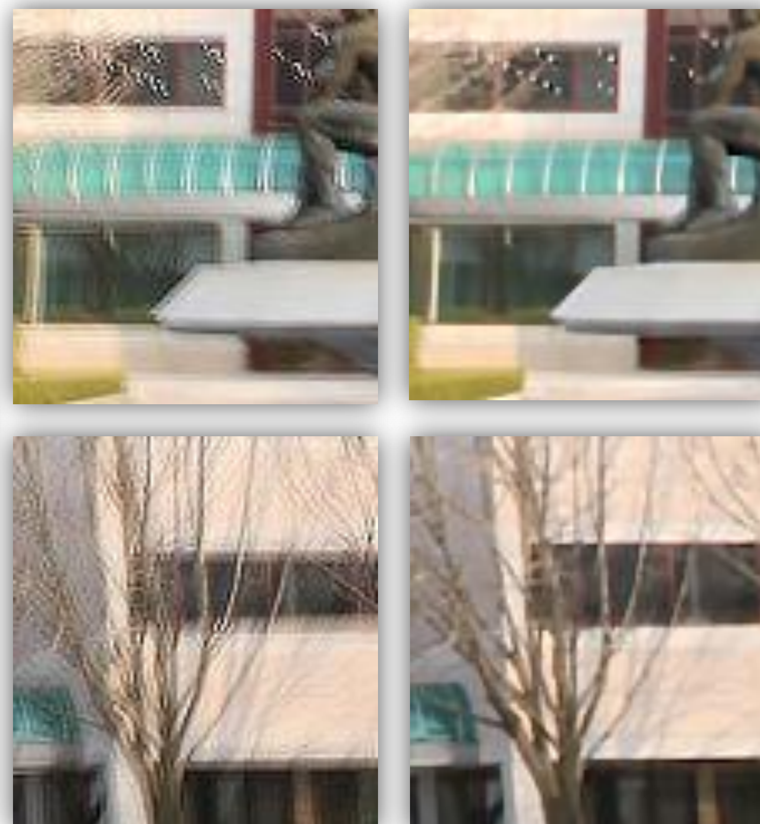


Non-uniform Blur





Our Method



[Shan et al. 2008] Our method

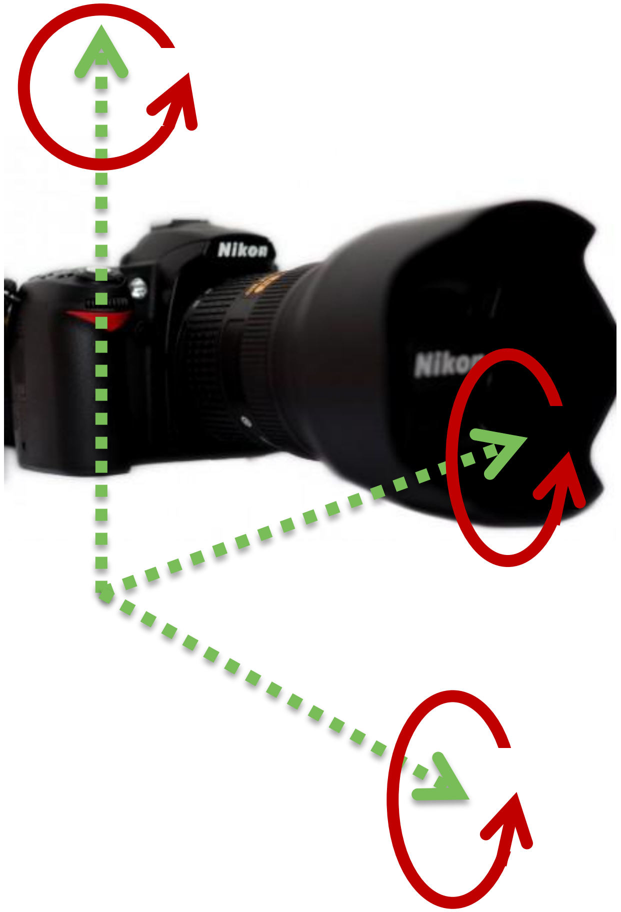
Related Work

SIGGRAPH2011

- Uniform motion blur
  - Fergus et al., SIGGRAPH 2006
  - Jia, CVPR 2007
  - Shan et al., SIGGRAPH 2008
  - Cho and Lee, SIGGRAPH ASIA 2009
  - Etc…

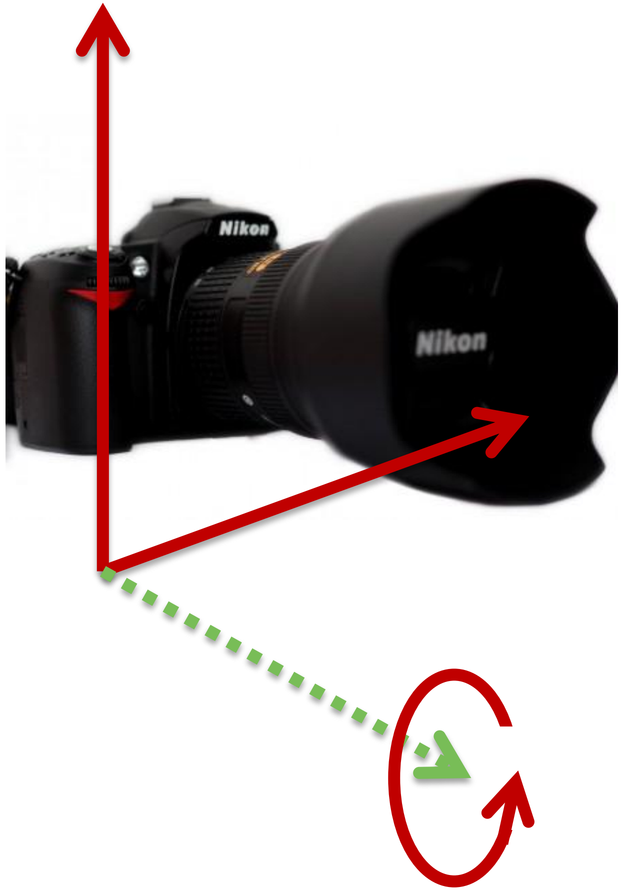
Related Work

- *Non-uniform* motion blur
  - Whyte et al. CVPR 2010
    - x, y, z rotations



Related Work

- *Non-uniform* motion blur
  - Whyte et al. CVPR 2010
    - x, y, z rotations
  - Gupta et al. ECCV 2010
    - x, y translations
      + in-plane rotation

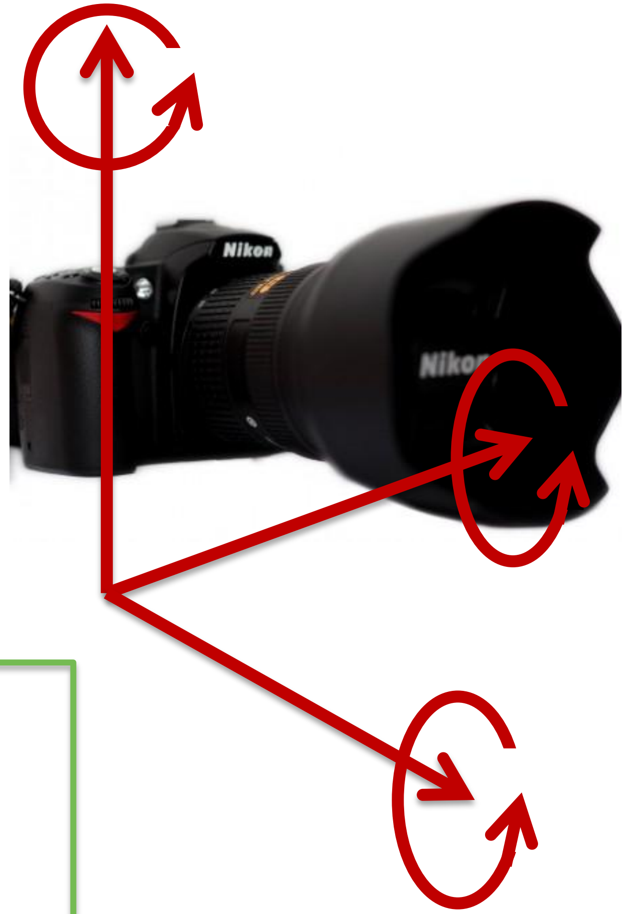


Related Work

- *Non-uniform* motion blur
  - Whyte et al. CVPR 2010
    - x, y, z rotations
  - Gupta et al. ECCV 2010
    - x, y translations
      + in-plane rotation

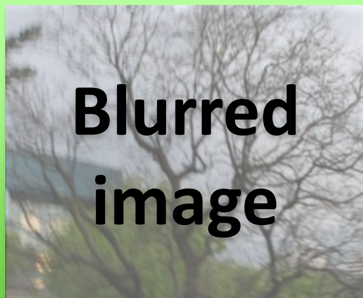
Our method

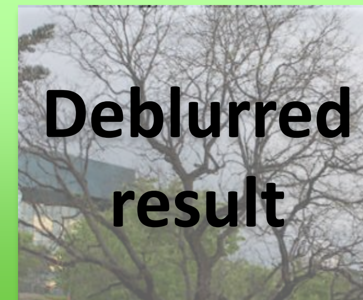
- x, y, z translations
  + x, y, z rotations



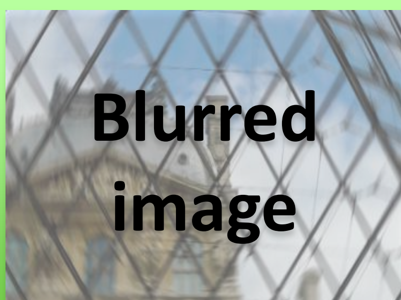
Contributions

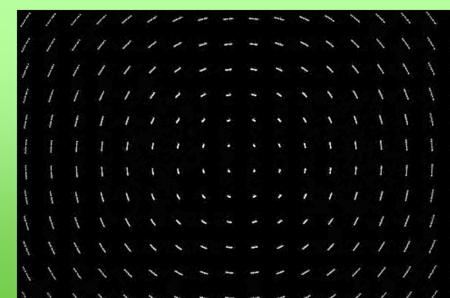
- Non-uniform motion deblurring



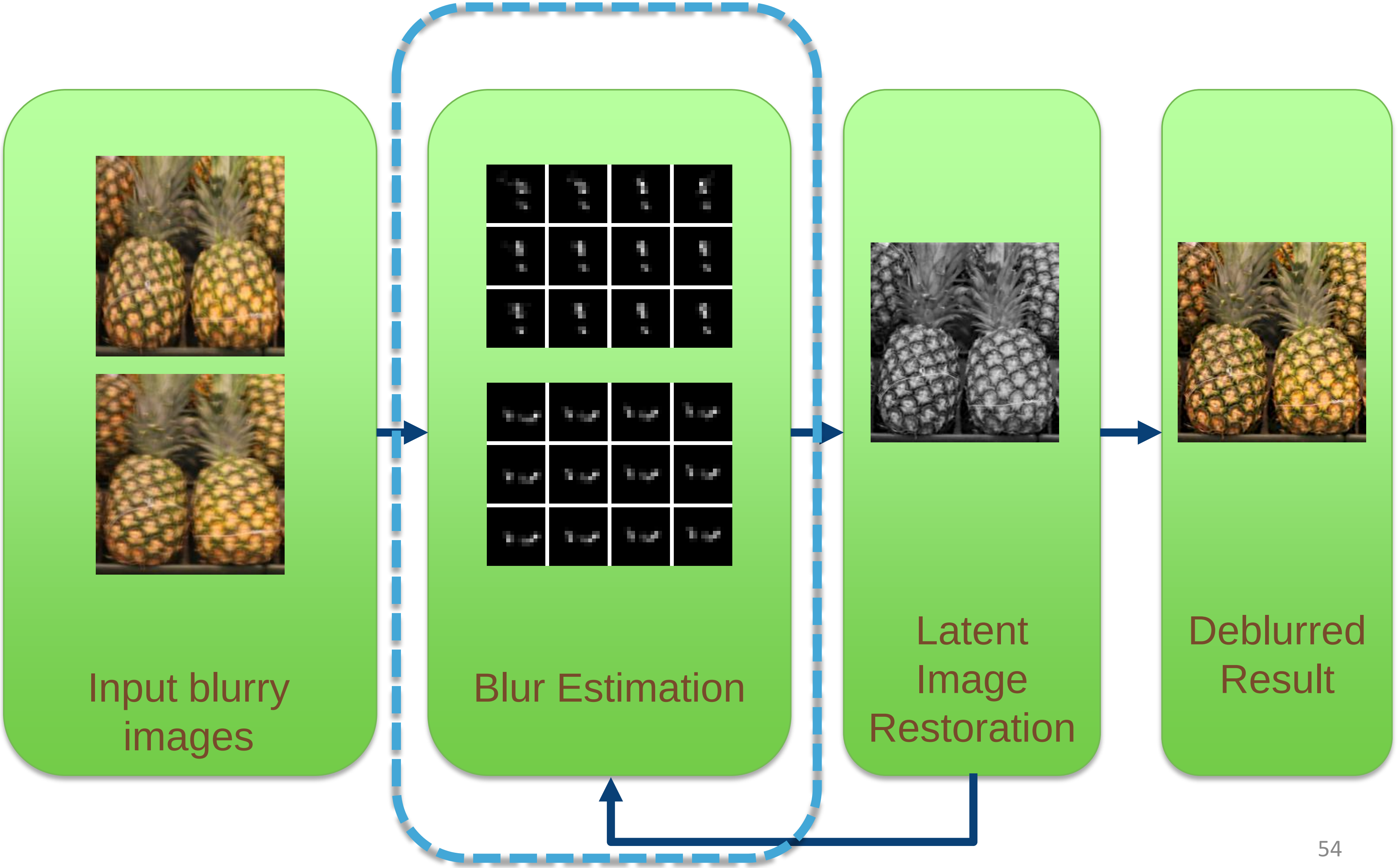


- Non-uniform blur estimation





Blind Deblurring

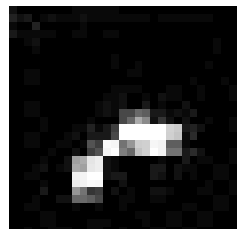


Uniform Blur Model

- Widely used in previous works

Blurred image

=



Motion blur kernel

*

Convolution operator

Latent image

Uniform Blur Model

- Widely used in previous works

Blurred image

$$= \sum_{i=1}^{N} \underbrace{w_i}_{\text{weight}} \underbrace{T_i}_{\text{translation}} \Big(\qquad \Big)$$

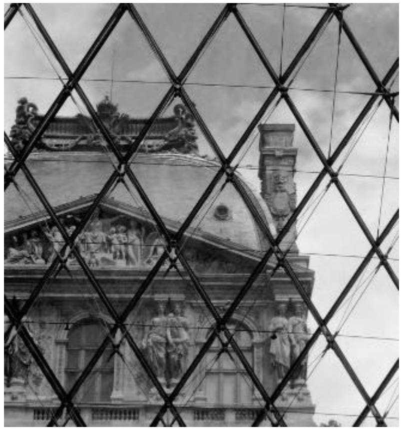
Latent image

Non-uniform Blur Model

- Tai et al., PAMI, to appear

Blurred image

$$= \sum_{i=1}^{N} w_i \underbrace{P_i}_{\text{Homography}} \Big(\quad \Big)$$

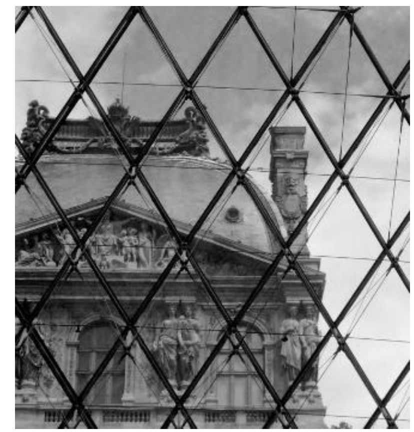


Latent image

Non-uniform Blur Model

- Tai et al., PAMI, to appear

$$= \sum_{i=1}^{N} \underline{w_i P_i} \Big($$



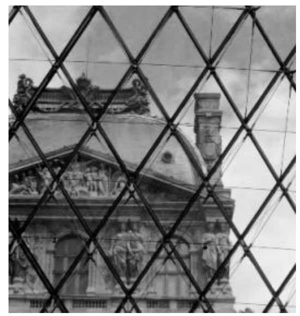
$$\Big)$$

How to estimate these?

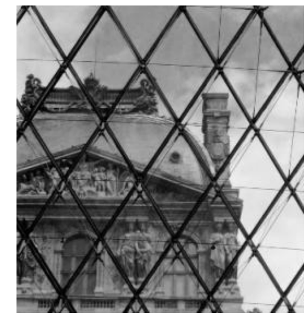
Blur Estimation

- Homography Estimation

$$- \sum_{j \neq i} w_j P_j \Big($$



$$\Big) = w_i P_i \Big($$



$$\Big)$$

$$= P_i \Big($$

$$\Big)$$

Blur Estimation

- Homography Estimation

$$= P_i($$

$$)$$

- Estimating P_i → Image registration problem
  - Lucas-Kanade based registration method

Blur Estimation

SIGGRAPH2011

- Homography Estimation

For $iter = 1$ to N_iters

For $i = 1$ to N

Fix other homographies, and

run a Lucas-Kanade method for solving

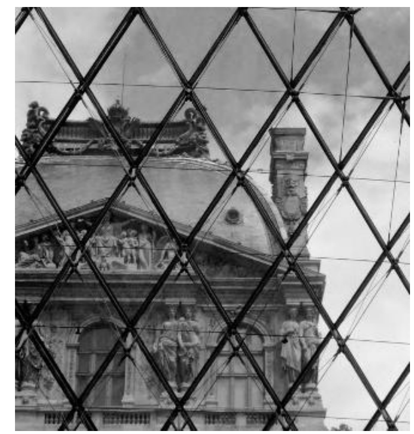
$= P_i($

$)$

Blur Estimation

- Weight Estimation: **simple linear system**

$$= \sum_{i=1}^{N} w_i P_i \Big(\quad \Big)$$



Blur Estimation

Input

→

Homography estimation

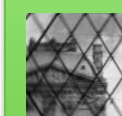
For each i, estimate P_i

 $= \boldsymbol{P}_i($  $)$

→

Weight estimation

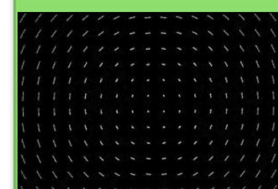
Solve a linear system

 $= \sum_{i=1}^{N} \boldsymbol{w}_i P_i \left(\right.$  $\left. \right)$

→

Output

Blur info: A set of P_i & w_i

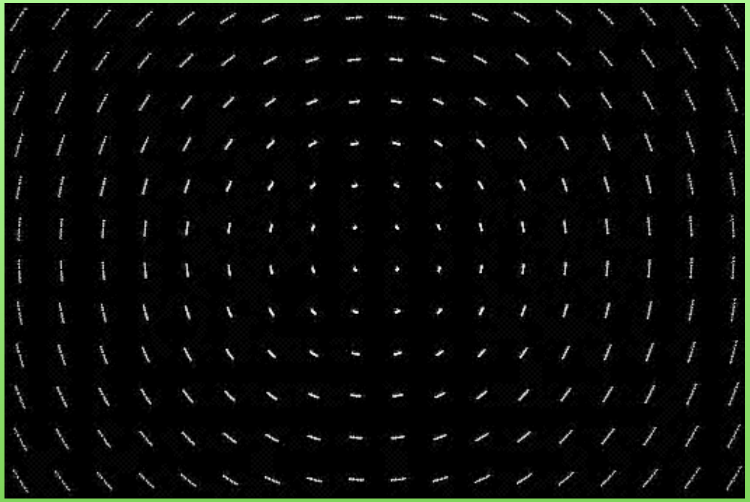


Blur Estimation

SIGGRAPH2011

Input

Estimated blur

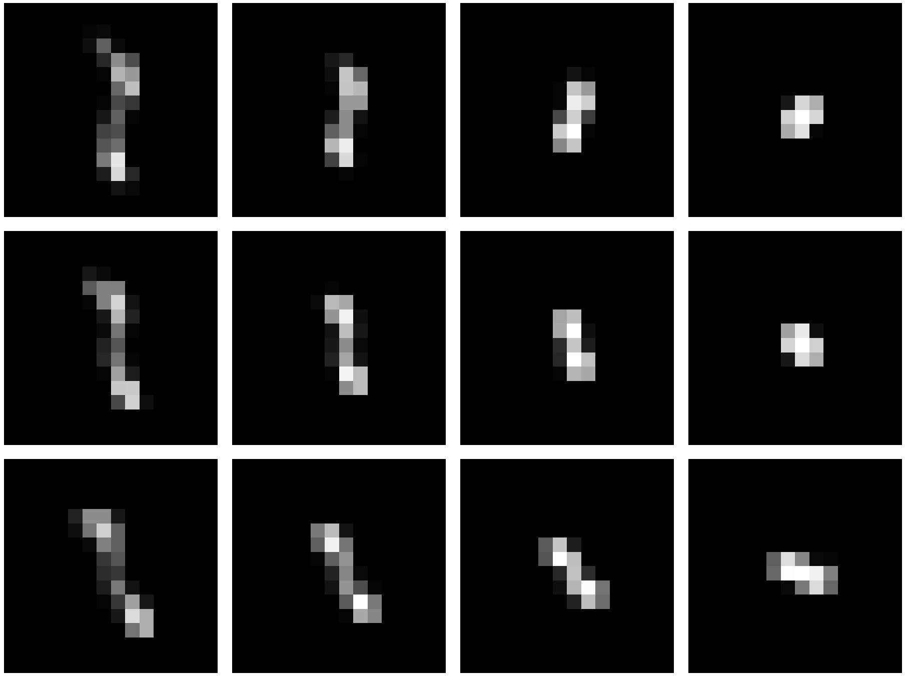


Results

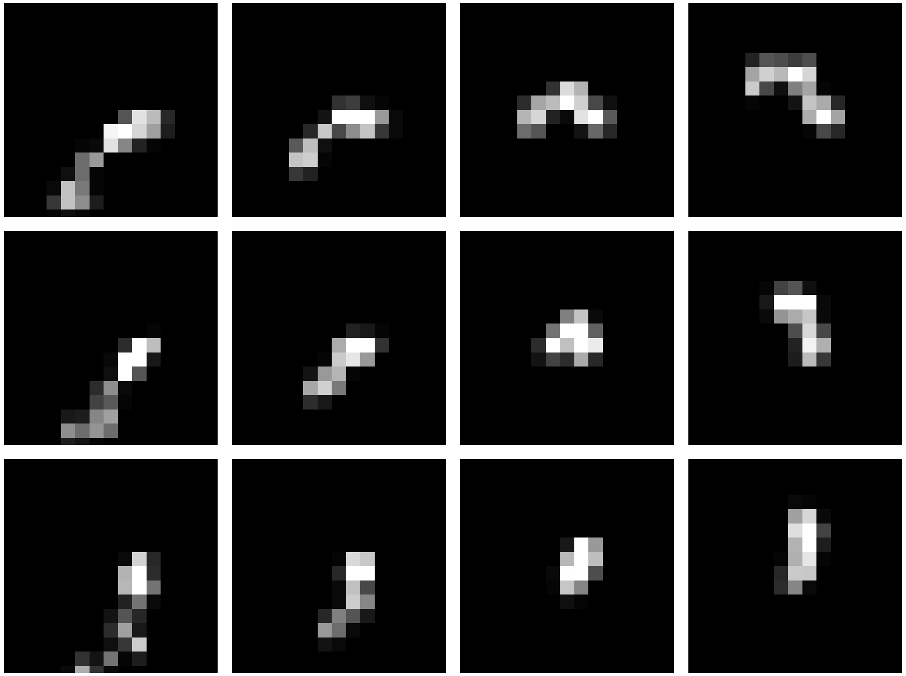
Blurred input 1

Blurred input 2

Results



Estimated blur 1



Estimated blur 2

Results

Uniform deblurring
[Cho and Lee 2009]

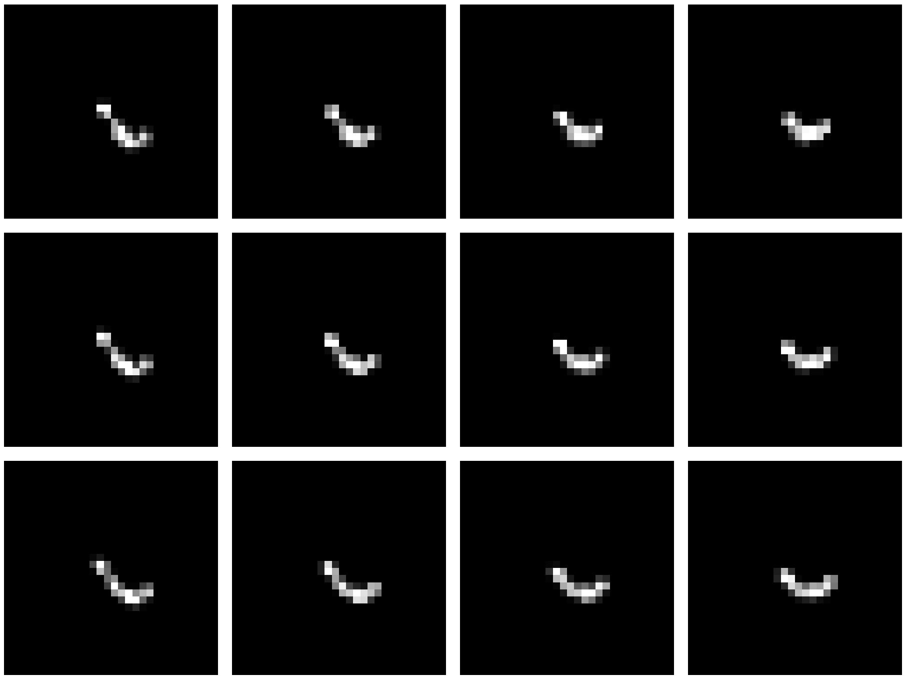
Our method

Results

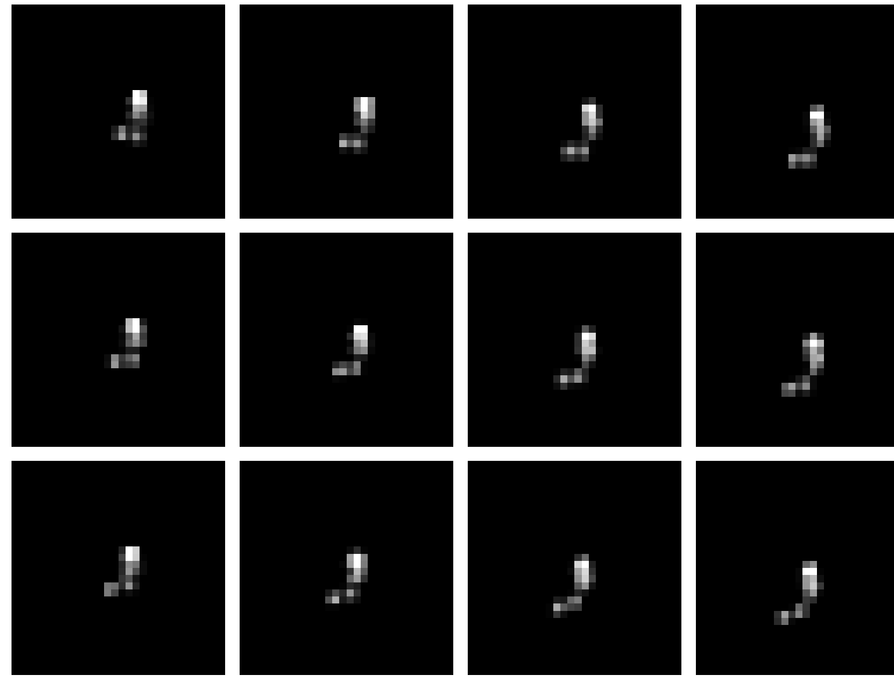
Blurred input 1

Blurred input 2

Results

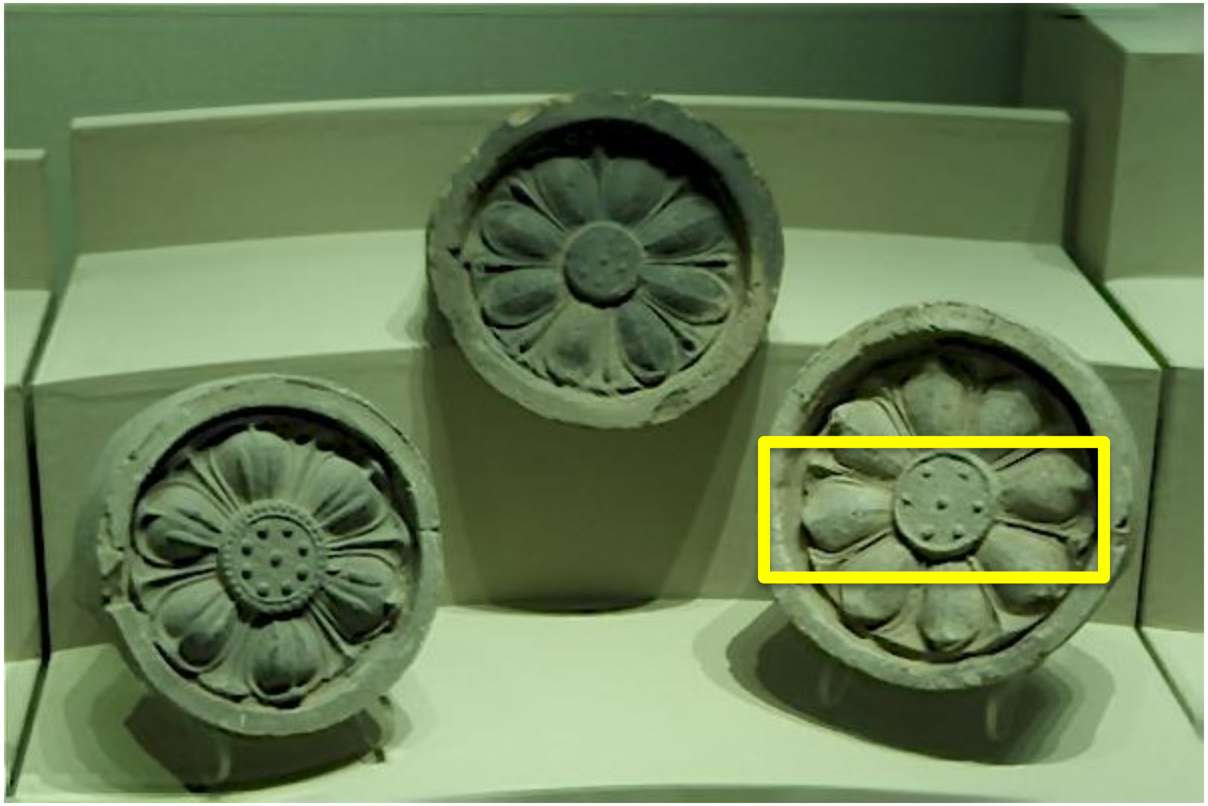


Estimated blur 1



Estimated blur 2

Results



Blurred input 1

Blurred input 2

Output

Results

Results

Results

Results

Results

Results

Conclusion

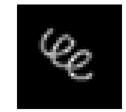
- We have developed
  - *Non-uniform* blur estimation method
  - Blind deblurring system for *non-uniform* motion blur

Future Work

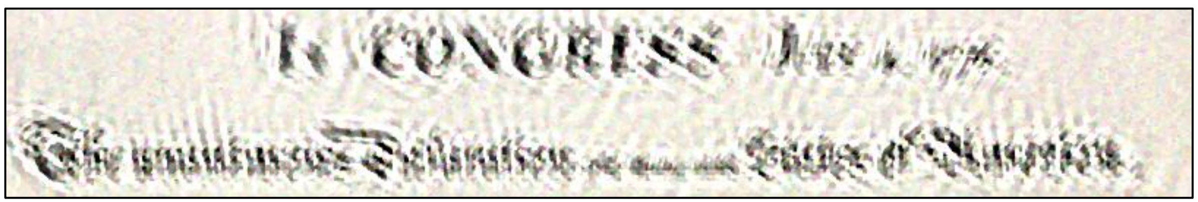
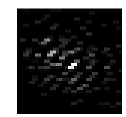
- Still ongoing…
- Analysis and comparison
  - Recent non-uniform deblurring methods
- More application
  - Video deblurring
    - More severe blur
    - Zooming blur

Text Deblurring

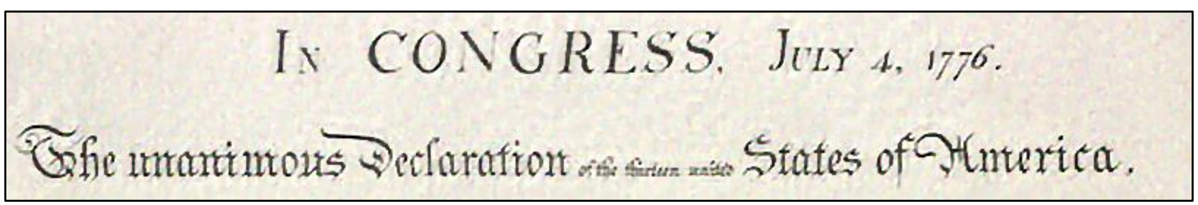
Original image

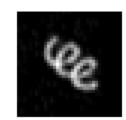



Synthetic blurred image

Fast motion deblurring result

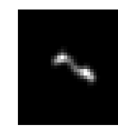




Our result

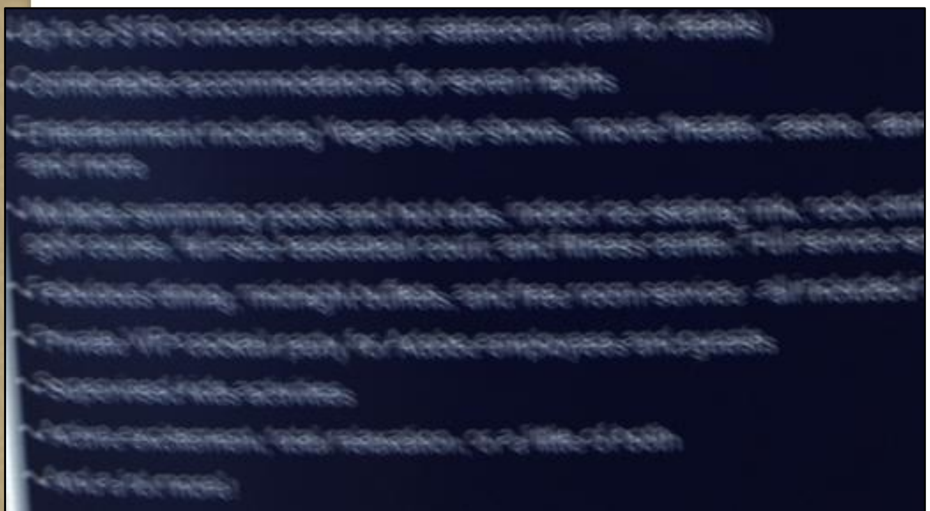
Results on Real Photographs (1/6)

Blurred image

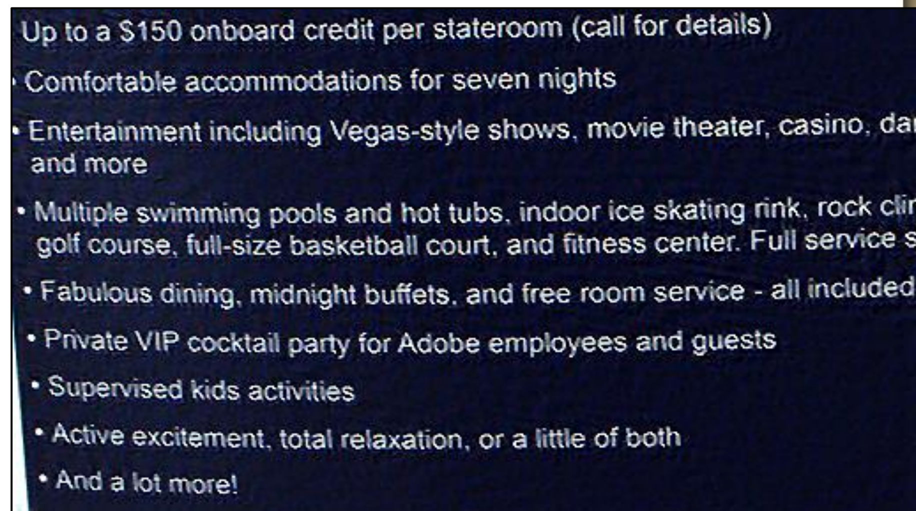


Our result

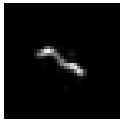
Results on Real Photographs (2/6)



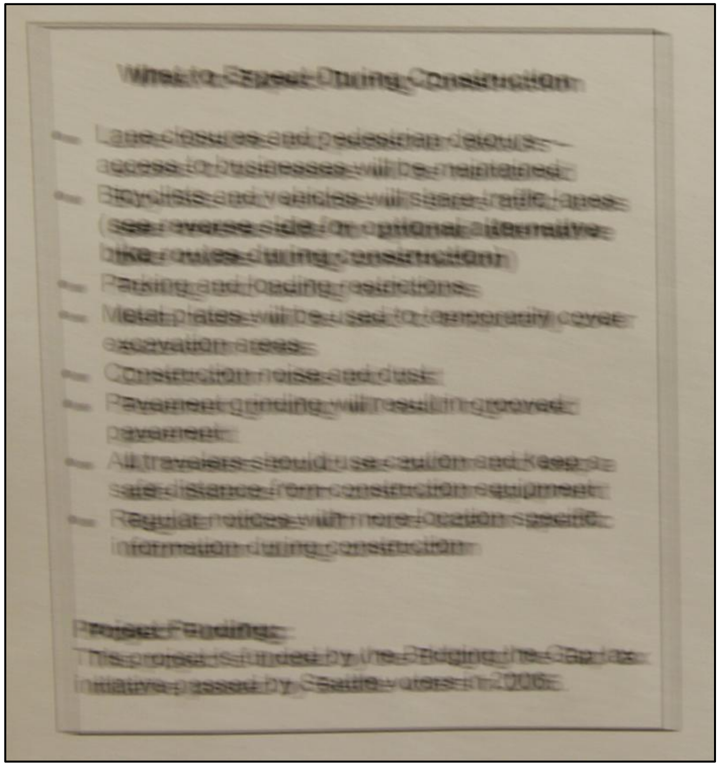
Blurred image



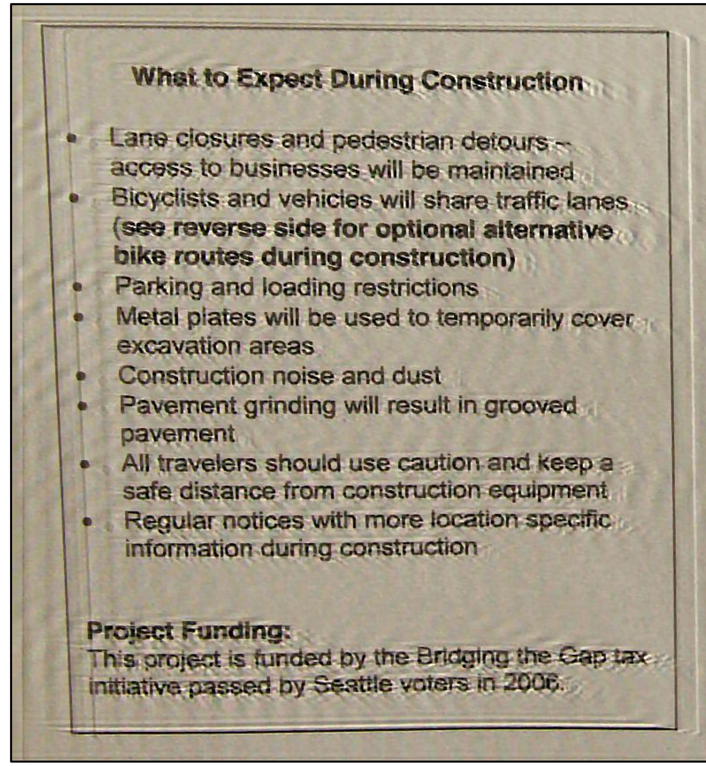
Our result



Results on Real Photographs (3/6)



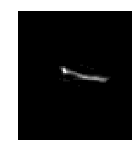
Blurred image



What to Expect During Construction

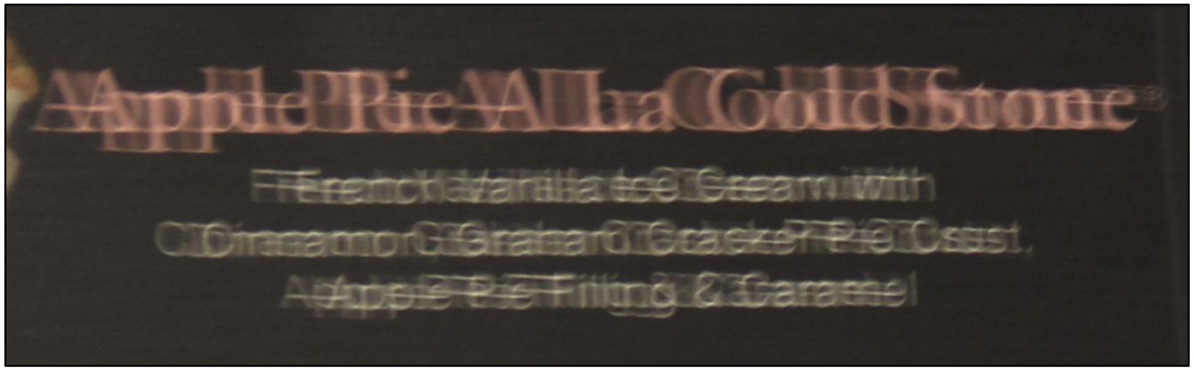
- Lane closures and pedestrian detours – access to businesses will be maintained
- Bicyclists and vehicles will share traffic lanes **(see reverse side for optional alternative bike routes during construction)**
- Parking and loading restrictions
- Metal plates will be used to temporarily cover excavation areas
- Construction noise and dust
- Pavement grinding will result in grooved pavement
- All travelers should use caution and keep a safe distance from construction equipment
- Regular notices with more location specific information during construction

Project Funding:
This project is funded by the Bridging the Gap tax initiative passed by Seattle voters in 2006.

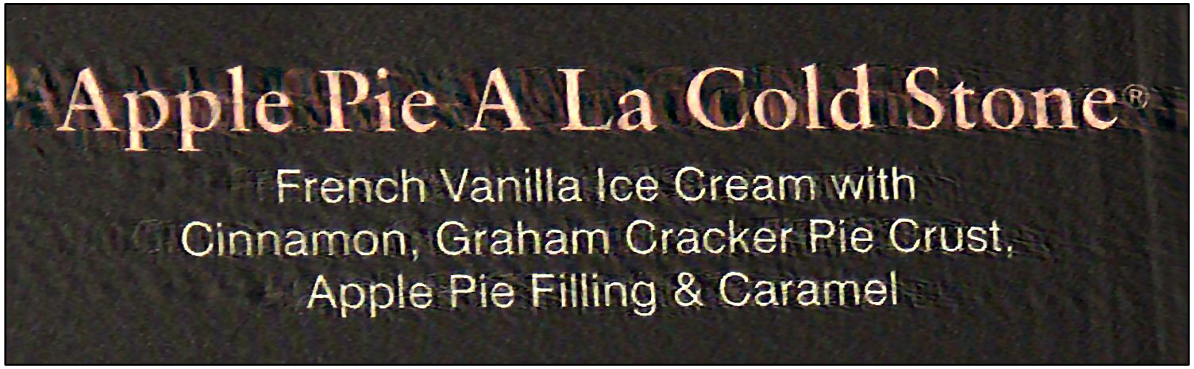


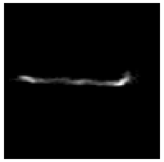
Our result

Results on Real Photographs (4/6)



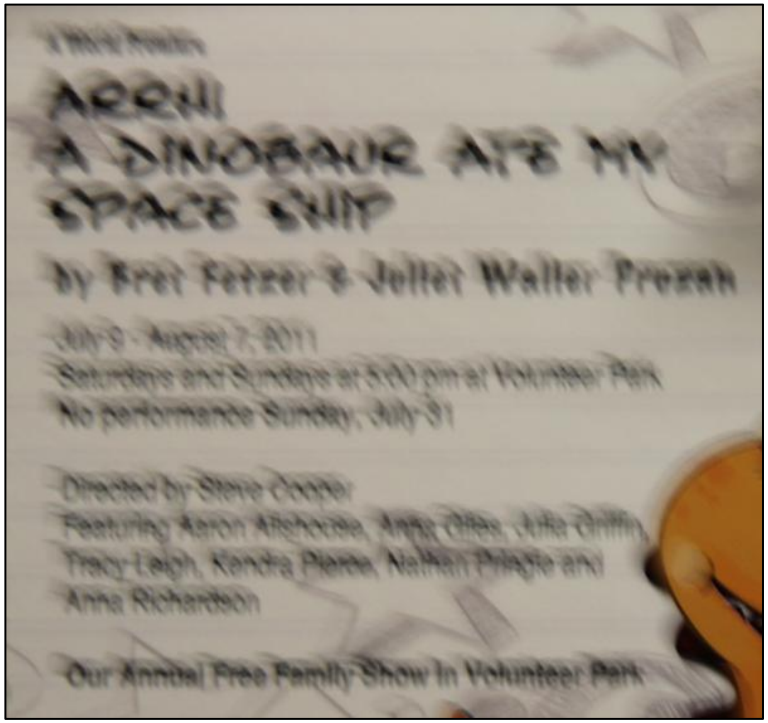
Blurred image



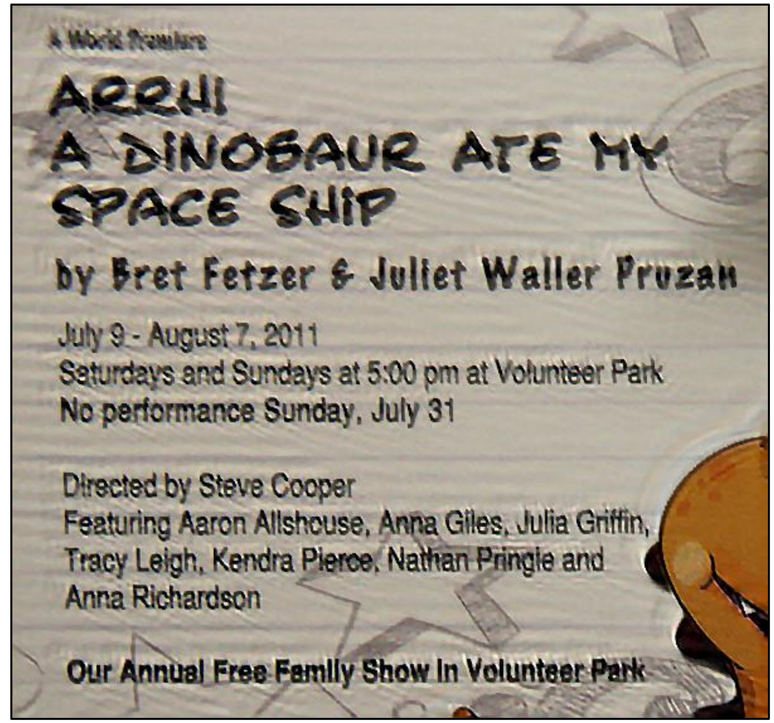


Our result (very large blur: 105x105)

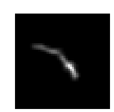
Results on Real Photographs (5/6)



Blurred image

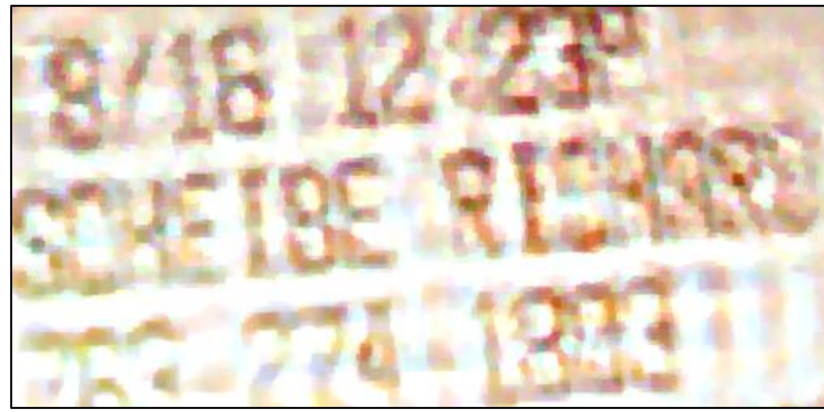


Our result

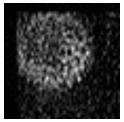


Results on Real Photographs (6/6)

Blurred image



Our result



Thank you!

http://cg.postech.ac.kr