

Learning High-Order Filters for Efficient Blind Deconvolution of Document Photographs

Lei Xiao, Jue Wang, Wolfgang Heidrich, Michael Hirsch

UNIVERSITY OF BRITISH COLUMBIA

ADOBE RESEARCH

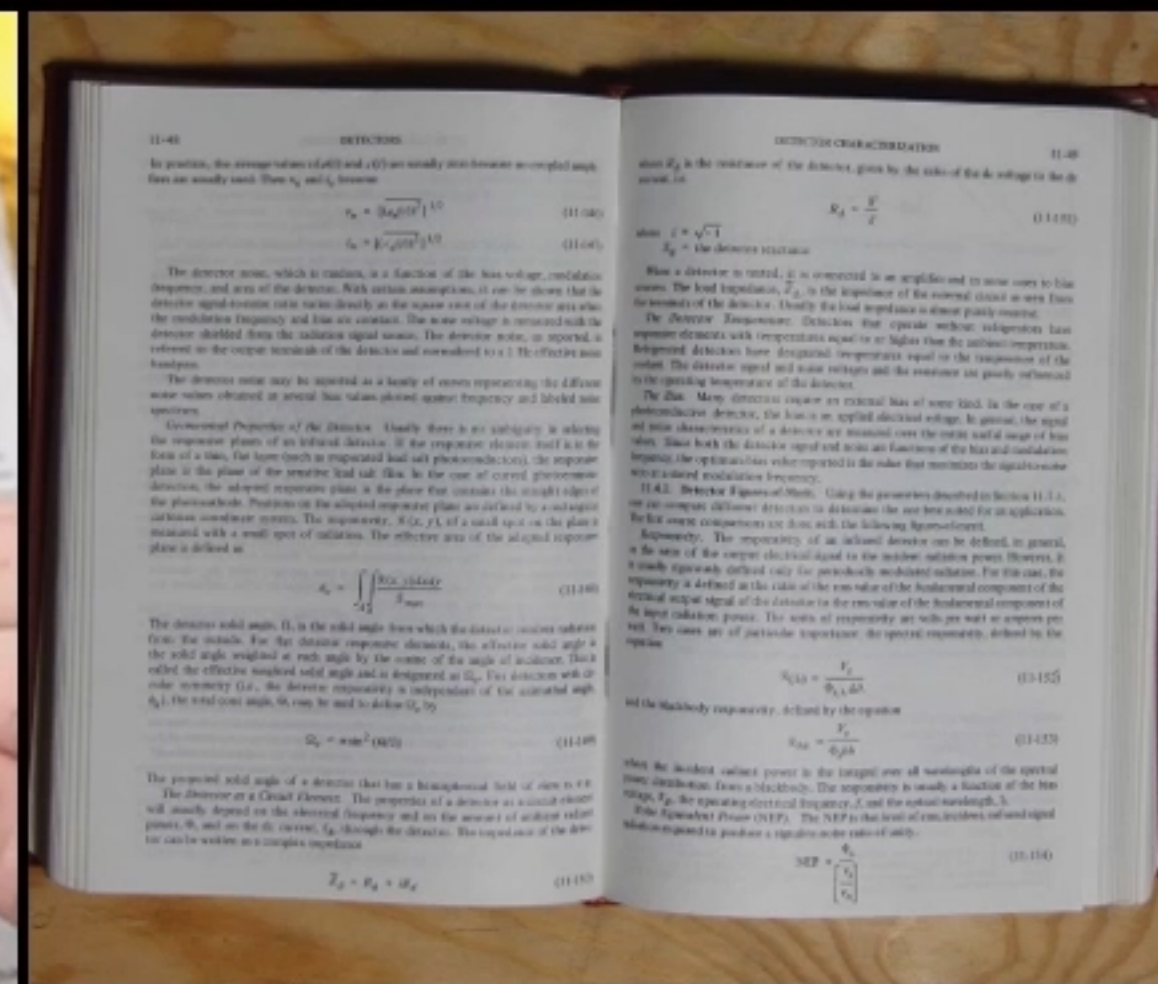
KAUST

MAX PLANCK INSTITUTE FOR INTELLIGENT SYSTEMS

Increasingly common to take photos of text documents



popularity of mobile cameras



printed articles, receipts, newspapers, books, etc

But fragile to blur caused by camera shake at hand-held shots

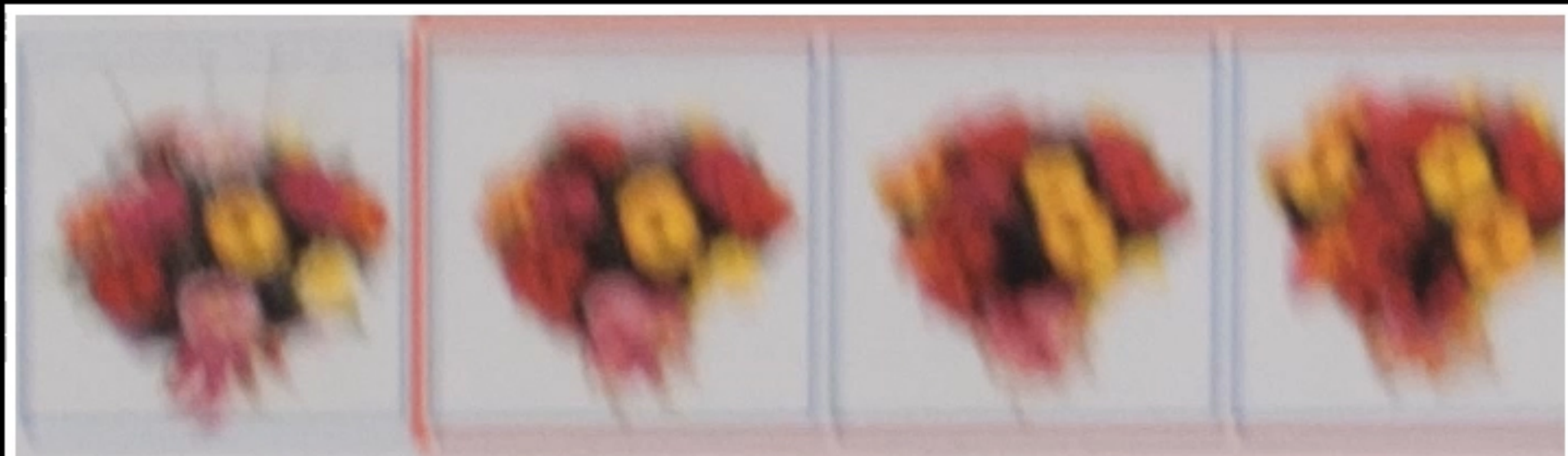


Figure 1. Regenerative morphing results of a clove and one flower bouquet into another. Both are / In the first example, it is hard to define (even / of correspondences between the source images, / traditional morphing. The resulting morphs ev / from one source image to the other without loss / appearance. Note the non-trivial transitions: facial / mouths) deform and merge into close-by similar / and similar flowers move towards each other whi

patch from an example input

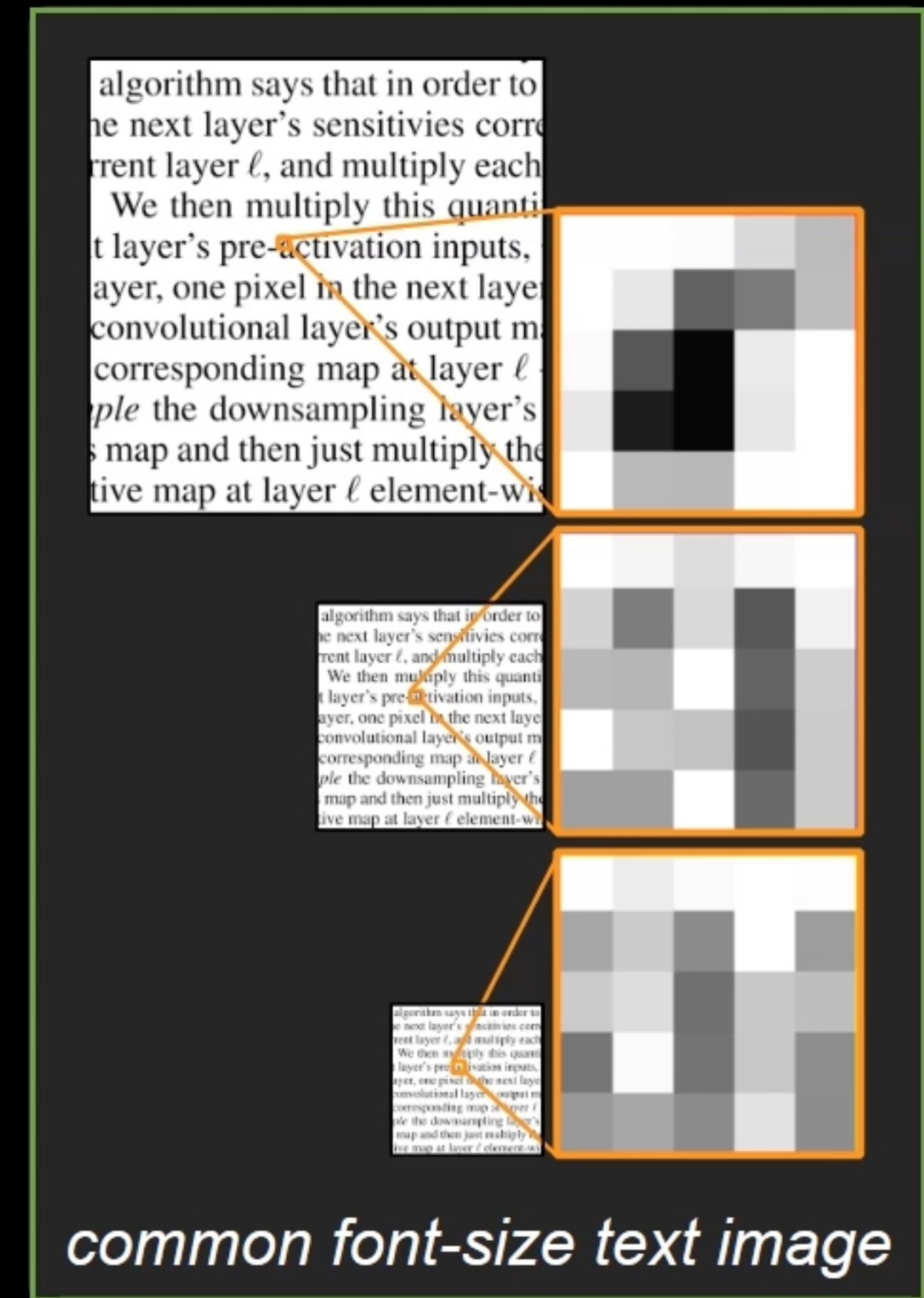
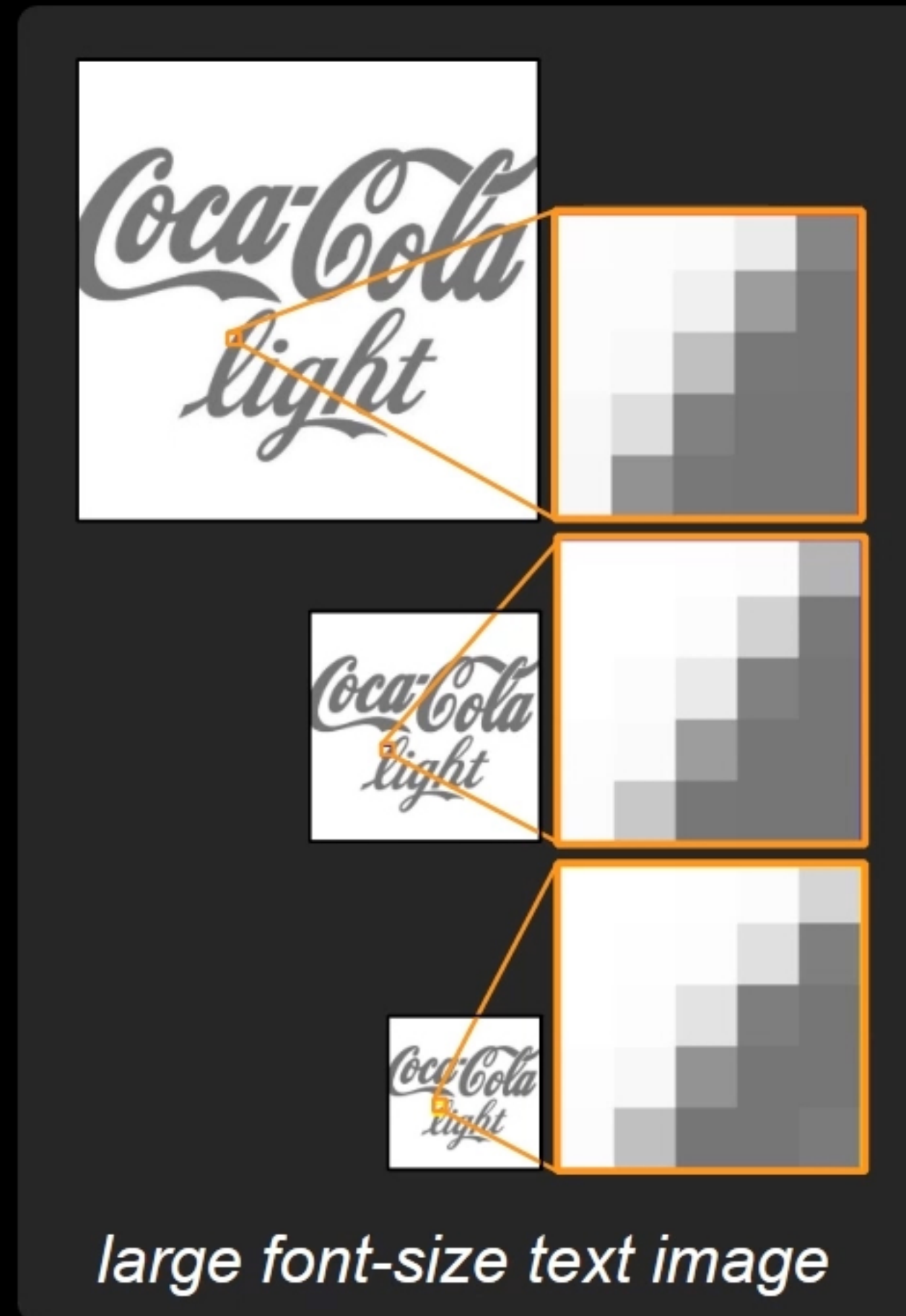
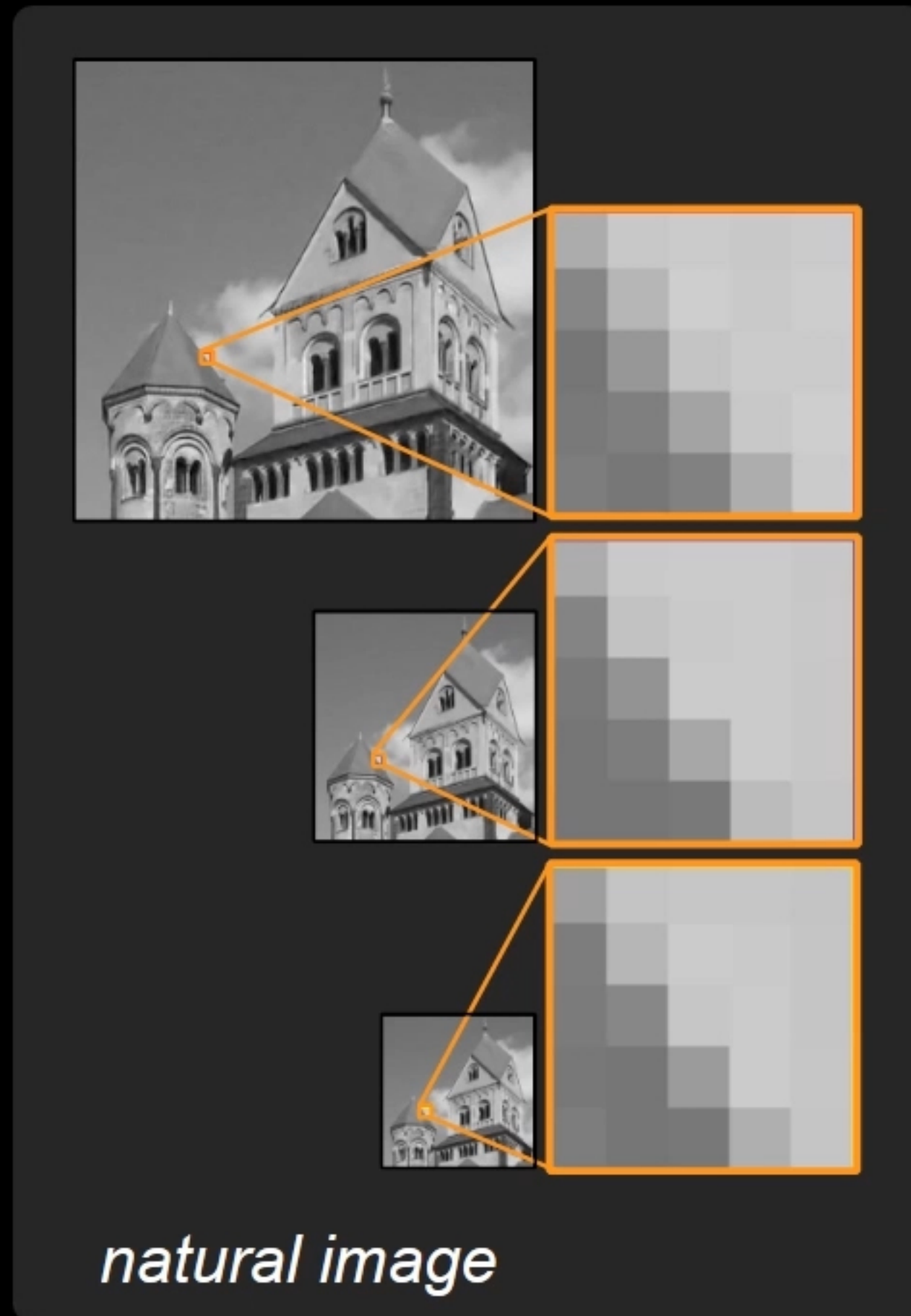
Blind Deconvolution



sharp and legible image

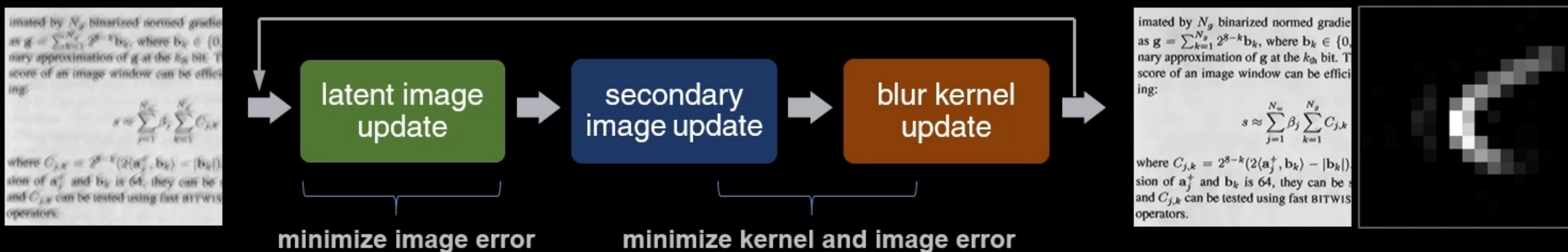
Sparse gradient priors do not work well for common text documents (typically w/ small fonts)

- Patch comparison at varying scales



Our solution: high-order filter priors learned via discriminative learning

- Multi-scale, interleaved shrinkage-fields network:

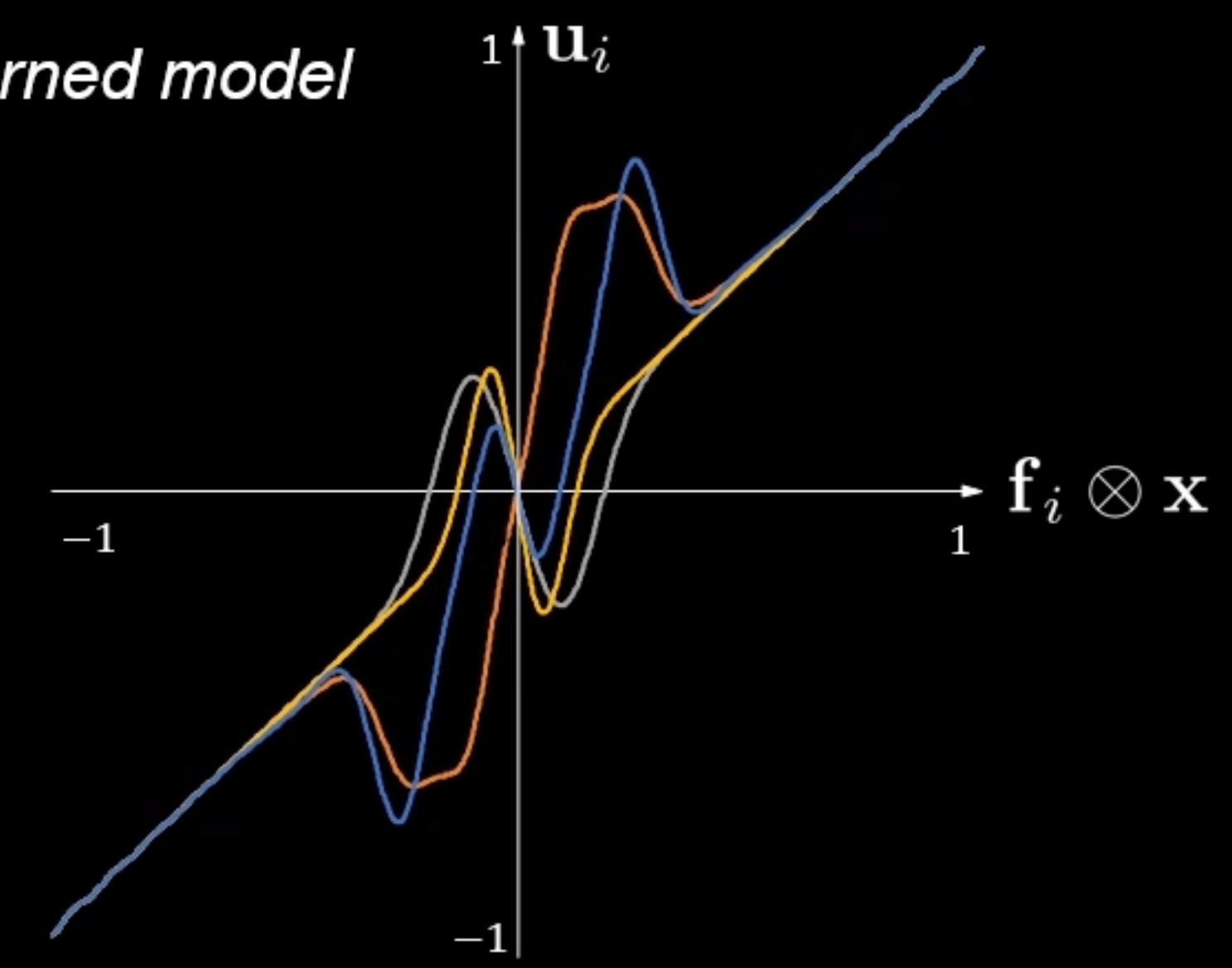
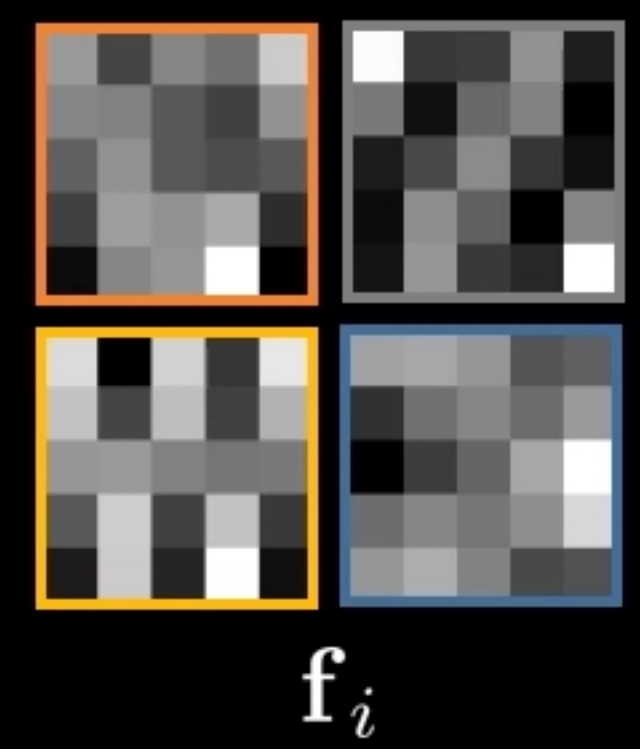


- Image update step:

$$\mathbf{x} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{b} - \mathbf{k} \otimes \mathbf{x}\|_2^2 + \sum_{i=1}^N \alpha \|\mathbf{u}_i - \mathbf{f}_i \otimes \mathbf{x}\|_2^2$$

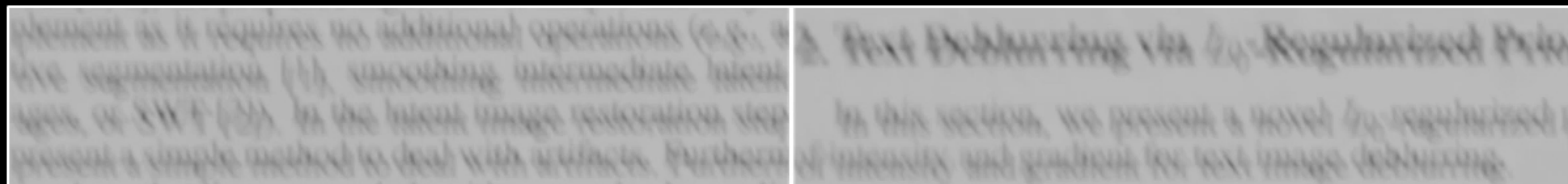
$$\mathbf{u}_i = \psi_i(\mathbf{f}_i \otimes \mathbf{x}) \quad [\text{Shrinkage-Field, CVPR14}]$$

examples of learned model

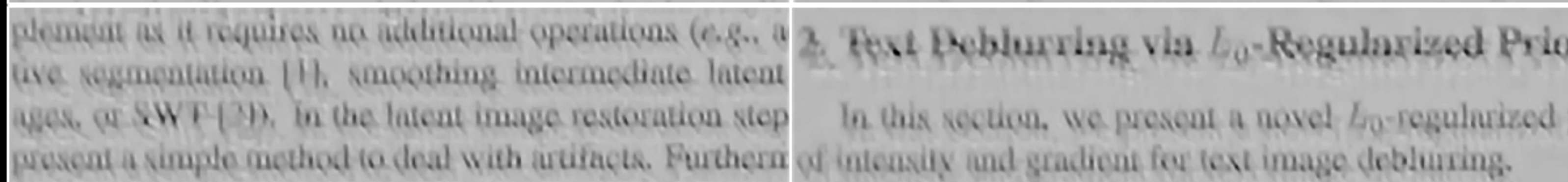


Results on real-world examples

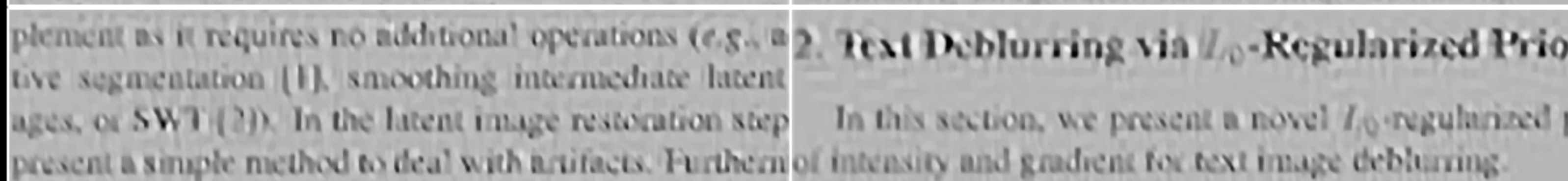
blurry input



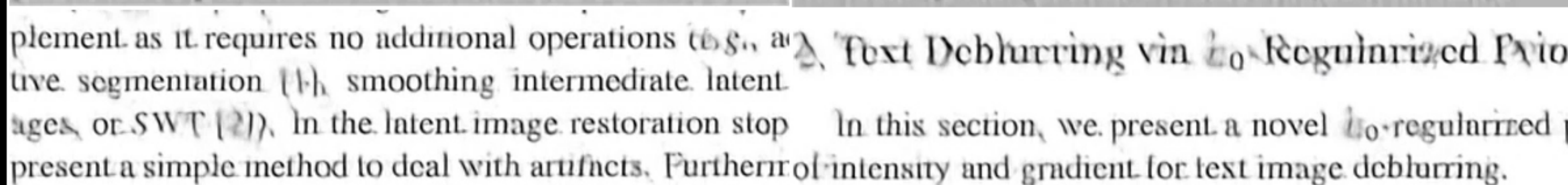
Xu (CVPR13)



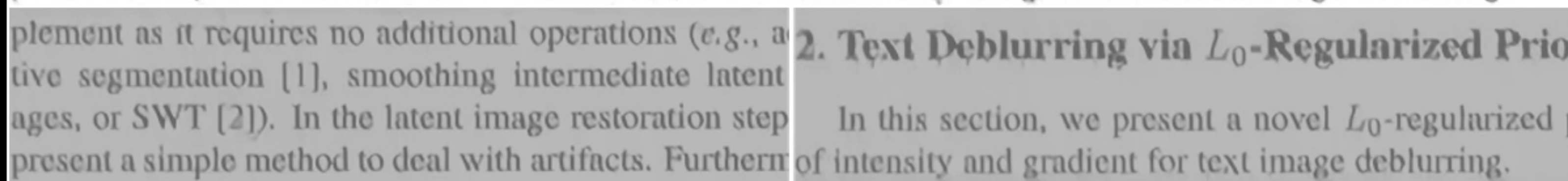
Pan (CVPR14)



Hradiš (BMVC15)



our result



example patches cropped from result images

Results on real-world examples

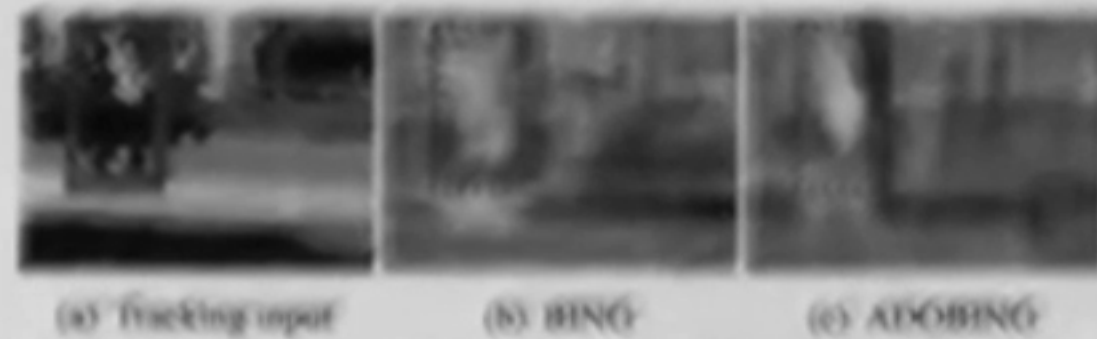


Figure 3. Objectness (BING) and adaptive objectness (ADOBING) for a specific tracking task. (a) The first frame with the initialization of the tracking target (red bounding box). (b) the objectness map of BING. (c) The objectness map of ADOBING.

imated by N_g binarized normed gradients (BING) feature as $g = \sum_{k=1}^{N_g} 2^{s-k} b_k$, where $b_k \in \{0, 1\}^{64}$ and is the binary approximation of g at the k_{th} bit. Then, the confidence score of an image window can be efficiently estimated using:

$$s \approx \sum_{j=1}^{N_w} \beta_j \sum_{k=1}^{N_g} C_{j,k}$$

where $C_{j,k} = 2^{s-k} (2 \langle \mathbf{a}_j^+, \mathbf{b}_k \rangle - |\mathbf{b}_k|)$. Since the dimension of \mathbf{a}_j^+ and \mathbf{b}_k is 64, they can be stored with INT64, and $C_{j,k}$ can be tested using fast BITWISE and POPCNT SSE operators.

3.2.2 Learning Adaptive Objectness

As briefed in Sec. 3.1, we can formulate the learning of adaptive objectness as following: Given training data $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ and a previously learned $\hat{w} \in \mathbb{R}^{64}$ ($i.e.$ BING), where $x_i \in \mathbb{R}^{64}$ is the

Algorithm 1 Coordinate Descent

Input: $w^{(1)} \in \mathbb{R}^n$
 1: **for** $k=1, 2, \dots$, iterate until ϵ
 2: $w^{(k,1)} = w^{(k)}$
 3: **for** $j = 1, 2, \dots, n$ **do**
 4: Find z by solving the sub-problem approximately.
 5: $w^{(k,j+1)} = w^{(k,j)} + z e_j$
 6: **end for**
 7: $w^{(k+1)} = w^{(k,n+1)}$
 8: **end for**
 9: **return** $w^{(k)}$

and $w^{(k,j)}$ for $w^{(k)}$ after update ($1 \leq j \leq 64$).

Following [9], for the i -th define $b_i(w) = 1 - y_i w^T x_i$. Then, a coordinate descent for the minimization of $w^{(k)}$ is achieved by solving the following n -dimensional sub-problem:

$$\min_z g_j(z) = |w_j^{(k,j)} - \hat{w}_j| + L_j(z; w^{(k)})$$

where subscript j indicates the j -th element of the vector, and $L_j(z; u) = C_j(z; u) + \lambda |z|$ and $e_j \in \mathbb{R}^{64}$ is the vector with all others be 0. The coordinate descent is summarized in Algorithm 1.

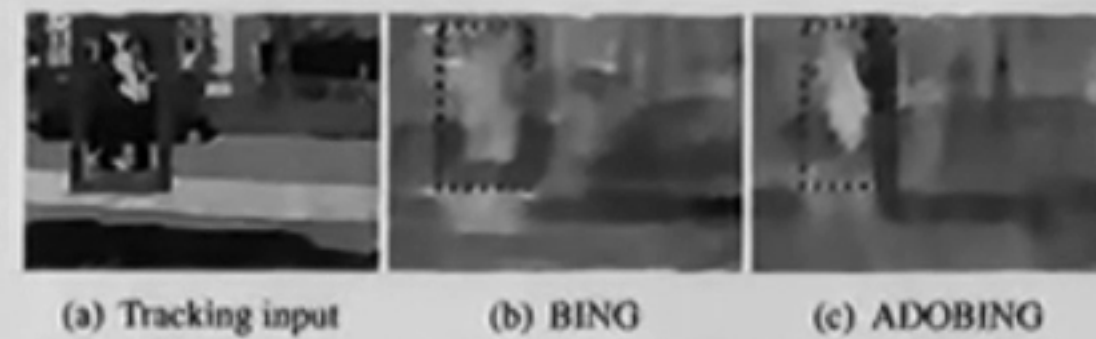


Figure 3. Objectness (BING) and adaptive objectness (ADOBING) for a specific tracking task. (a) The first frame with the initialization of the tracking target (red bounding box). (b) the objectness map of BING. (c) The objectness map of ADOBING.

imated by N_g binarized normed gradients (BING) feature as $g = \sum_{k=1}^{N_g} 2^{s-k} b_k$, where $b_k \in \{0, 1\}^{64}$ and is the binary approximation of g at the k_{th} bit. Then, the confidence score of an image window can be efficiently estimated using:

$$s \approx \sum_{j=1}^{N_w} \beta_j \sum_{k=1}^{N_g} C_{j,k}$$

where $C_{j,k} = 2^{s-k} (2 \langle \mathbf{a}_j^+, \mathbf{b}_k \rangle - |\mathbf{b}_k|)$. Since the dimension of \mathbf{a}_j^+ and \mathbf{b}_k is 64, they can be stored with INT64, and $C_{j,k}$ can be tested using fast BITWISE and POPCNT SSE operators.

3.2.2 Learning Adaptive Objectness

As briefed in Sec. 3.1, we can formulate the learning of adaptive objectness as following: Given training data $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ and a previously learned $\hat{w} \in \mathbb{R}^{64}$ ($i.e.$ BING), where $x_i \in \mathbb{R}^{64}$ is the

Algorithm 1 Coordinate Descent

Input: $w^{(1)} \in \mathbb{R}^n$
 1: **for** $k=1, 2, \dots$, iterate until ϵ
 2: $w^{(k,1)} = w^{(k)}$
 3: **for** $j = 1, 2, \dots, n$ **do**
 4: Find z by solving the sub-problem approximately.
 5: $w^{(k,j+1)} = w^{(k,j)} + z e_j$
 6: **end for**
 7: $w^{(k+1)} = w^{(k,n+1)}$
 8: **end for**
 9: **return** $w^{(k)}$

and $w^{(k,j)}$ for $w^{(k)}$ after update ($1 \leq j \leq 64$).

Following [9], for the i -th define $b_i(w) = 1 - y_i w^T x_i$. Then, a coordinate descent for the minimization of $w^{(k)}$ is achieved by solving the following n -dimensional sub-problem:

$$\min_z g_j(z) = |w_j^{(k,j)} - \hat{w}_j| + L_j(z; w^{(k)})$$

where subscript j indicates the j -th element of the vector, and $L_j(z; u) = C_j(z; u) + \lambda |z|$ and $e_j \in \mathbb{R}^{64}$ is the vector with all others be 0. The coordinate descent is summarized in Algorithm 1.

input

our result

pan and zoom animation

Results on spatially-varying blur

blurry input

4 Fast Latent Image Estimation

Prediction In the prediction step, we estimate the image gradient maps $\{P_x, P_y\}$ of the latent image L in which only the salient edges remain and other regions have zero gradients. Consequently, in the kernel estimation step, only the salient edges have influences on optimization of the kernel because convolution of zero gradients is always zero regardless of the kernel.

We use a shock filter to restore strong edges in L . A shock filter is an effective tool for enhancing image features, which can recover sharp edges from blurred step signals [Osher and Rudin 1990]. The evolution equation of a shock filter is formulated as

$$I_{t+1} = I_t - \text{sign}(\Delta I_t) \|\nabla I_t\| dt, \quad (4)$$

where I_t is an image at time t , and ΔI_t and ∇I_t are the Laplacian and gradient of I_t , respectively. dt is the time step for a single evolution.

Our prediction step consists of bilateral filtering, shock filtering,

quantized by 45° , and gradients of opposite directions are counted together. Then, we find a threshold that keeps at least rm pixels from the largest magnitude for each quantized angle. We use 2 for r by default. To include more gradient values in $\{P_x, P_y\}$ as the deblurring iteration progresses, we gradually decrease the threshold determined at the beginning by multiplying 0.9 at each iteration.

Deconvolution In the deconvolution step, we estimate the latent image L from a given kernel K and the input blurred image B . We use the energy function

$$f_L(L) = \sum_{\partial_*} \omega_* \|K * \partial_* L - \partial_* B\|^2 + \alpha \|\nabla L\|^2, \quad (8)$$

where $\partial_* \in \{\partial_o, \partial_x, \partial_y, \partial_{xx}, \partial_{xy}, \partial_{yy}\}$ denotes the partial derivative operator in different directions and orders, $\omega_* \in \{\omega_0, \omega_1, \omega_2\}$ is a weight for each partial derivative, and α is a weight for the regularization term. The first term in the energy is based on the blur model of [Shan et al. 2008], which uses image derivatives for re-

our result image

4 Fast Latent Image Estimation

Prediction In the prediction step, we estimate the image gradient maps $\{P_x, P_y\}$ of the latent image L in which only the salient edges remain and other regions have zero gradients. Consequently, in the kernel estimation step, only the salient edges have influences on optimization of the kernel because convolution of zero gradients is always zero regardless of the kernel.

We use a shock filter to restore strong edges in L . A shock filter is an effective tool for enhancing image features, which can recover sharp edges from blurred step signals [Osher and Rudin 1990]. The evolution equation of a shock filter is formulated as

$$I_{t+1} = I_t - \text{sign}(\Delta I_t) \|\nabla I_t\| dt, \quad (4)$$

where I_t is an image at time t , and ΔI_t and ∇I_t are the Laplacian and gradient of I_t , respectively. dt is the time step for a single evolution.

Our prediction step consists of bilateral filtering, shock filtering,

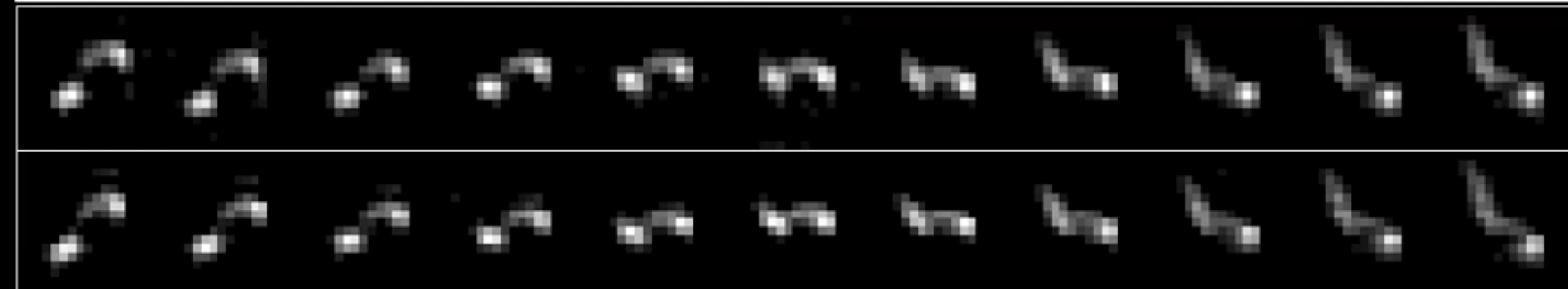
quantized by 45° , and gradients of opposite directions are counted together. Then, we find a threshold that keeps at least rm pixels from the largest magnitude for each quantized angle. We use 2 for r by default. To include more gradient values in $\{P_x, P_y\}$ as the deblurring iteration progresses, we gradually decrease the threshold determined at the beginning by multiplying 0.9 at each iteration.

Deconvolution In the deconvolution step, we estimate the latent image L from a given kernel K and the input blurred image B . We use the energy function

$$f_L(L) = \sum_{\partial_*} \omega_* \|K * \partial_* L - \partial_* B\|^2 + \alpha \|\nabla L\|^2, \quad (5)$$

where $\partial_* \in \{\partial_o, \partial_x, \partial_y, \partial_{xx}, \partial_{xy}, \partial_{yy}\}$ denotes the partial derivative operator in different directions and orders, $\omega_* \in \{\omega_0, \omega_1, \omega_2\}$ is a weight for each partial derivative, and α is a weight for the regularization term. The first term in the energy is based on the blur model of [Shan et al. 2008], which uses image derivatives for re-

our blur estimate



true blur



Results on documents containing color figures

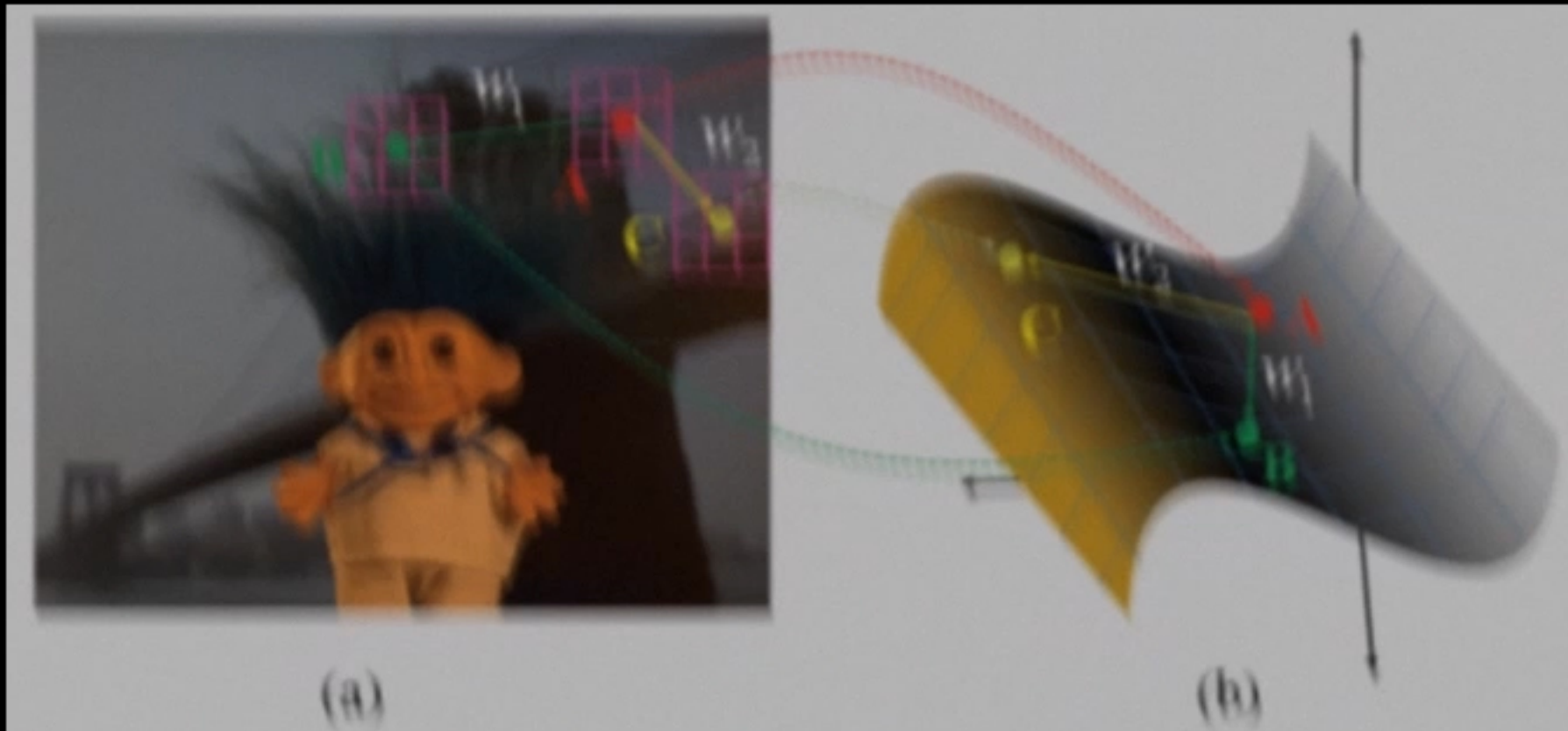


Figure 1. (a) the input image, (b) the corresponding feature space. The pixel A can be generated by linearly combining the color at B and C .

this method was extended by Chuang et al. [5] with the Bayesian estimation framework. These methods work well when the unknown pixels are near the foreground bound-

blurry input

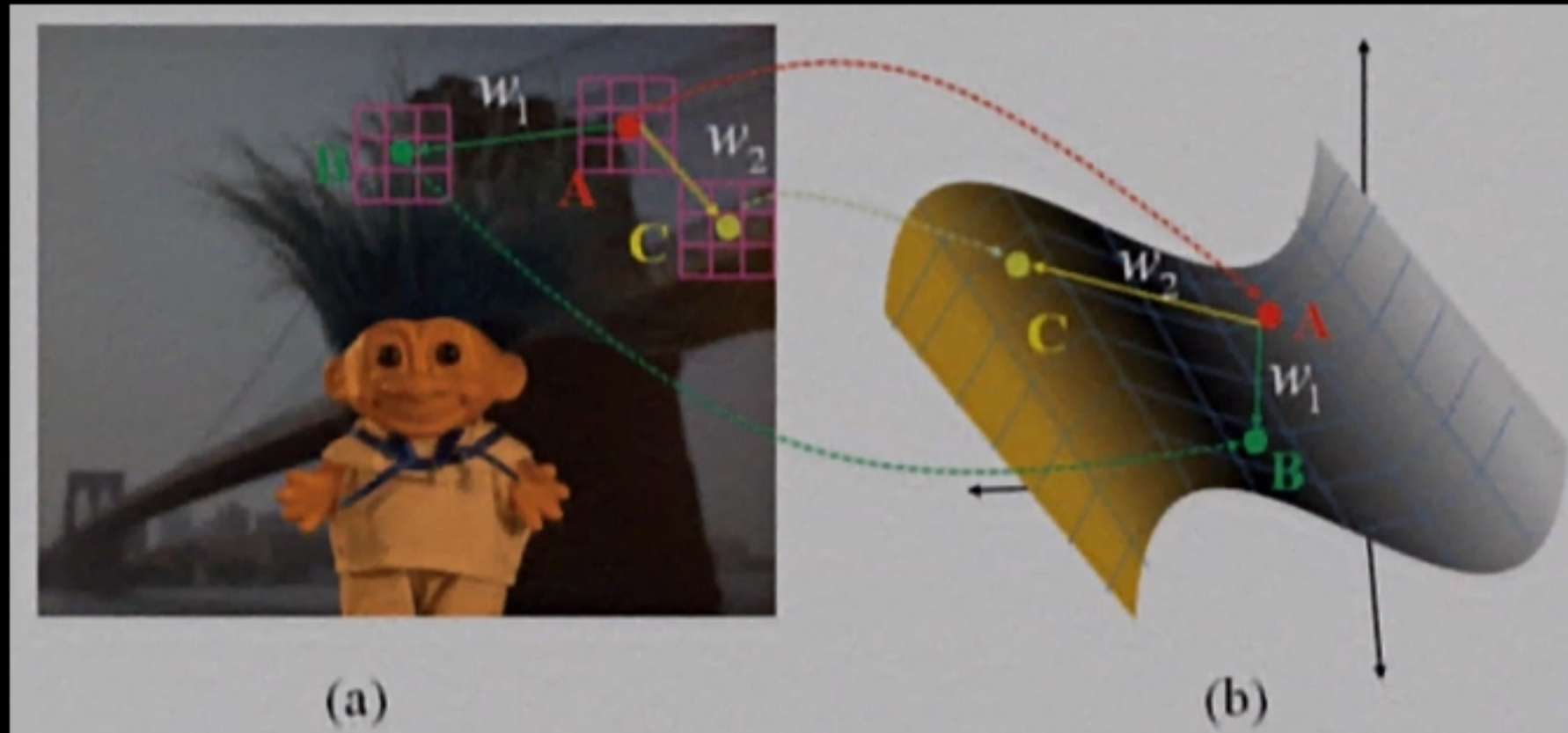


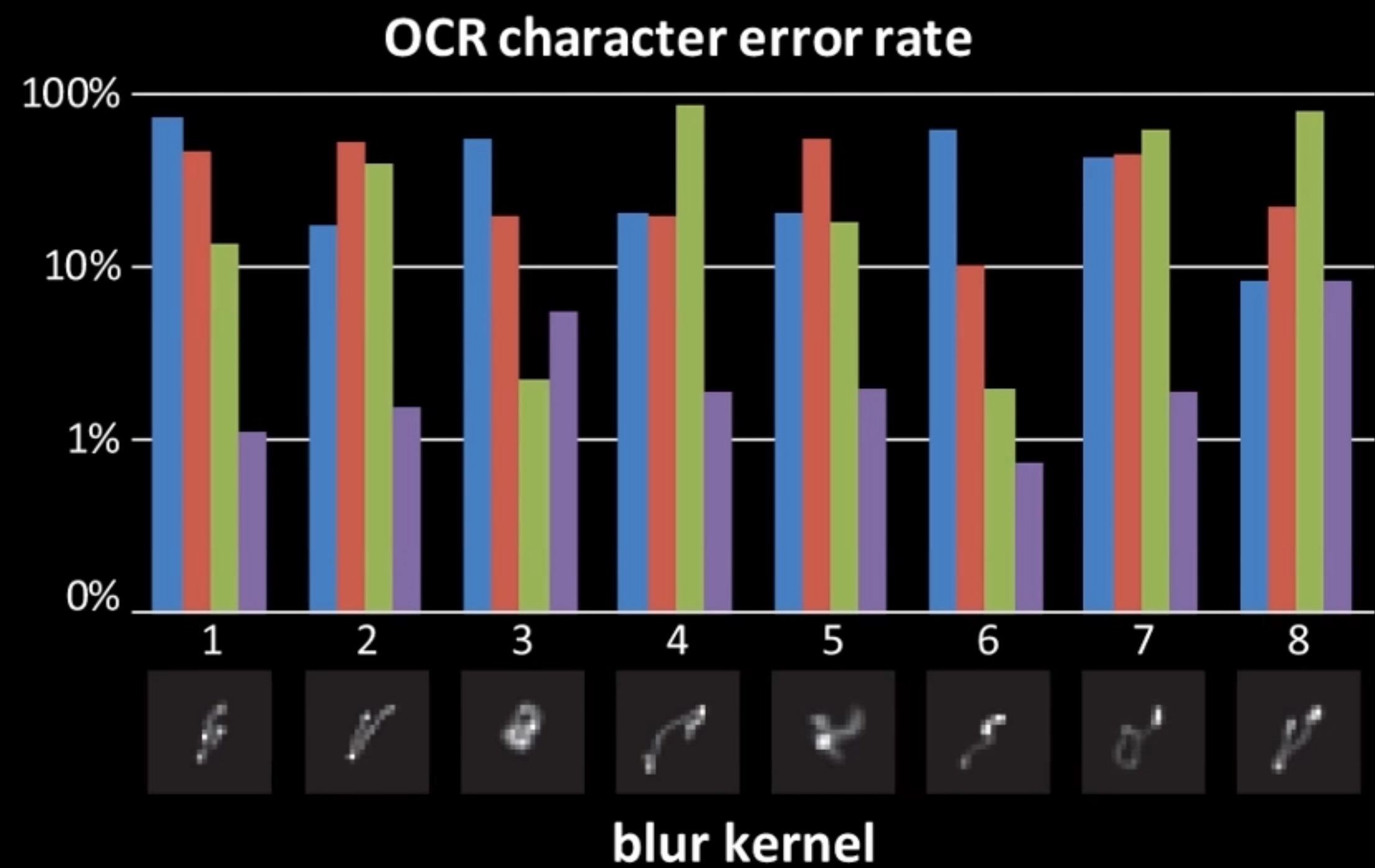
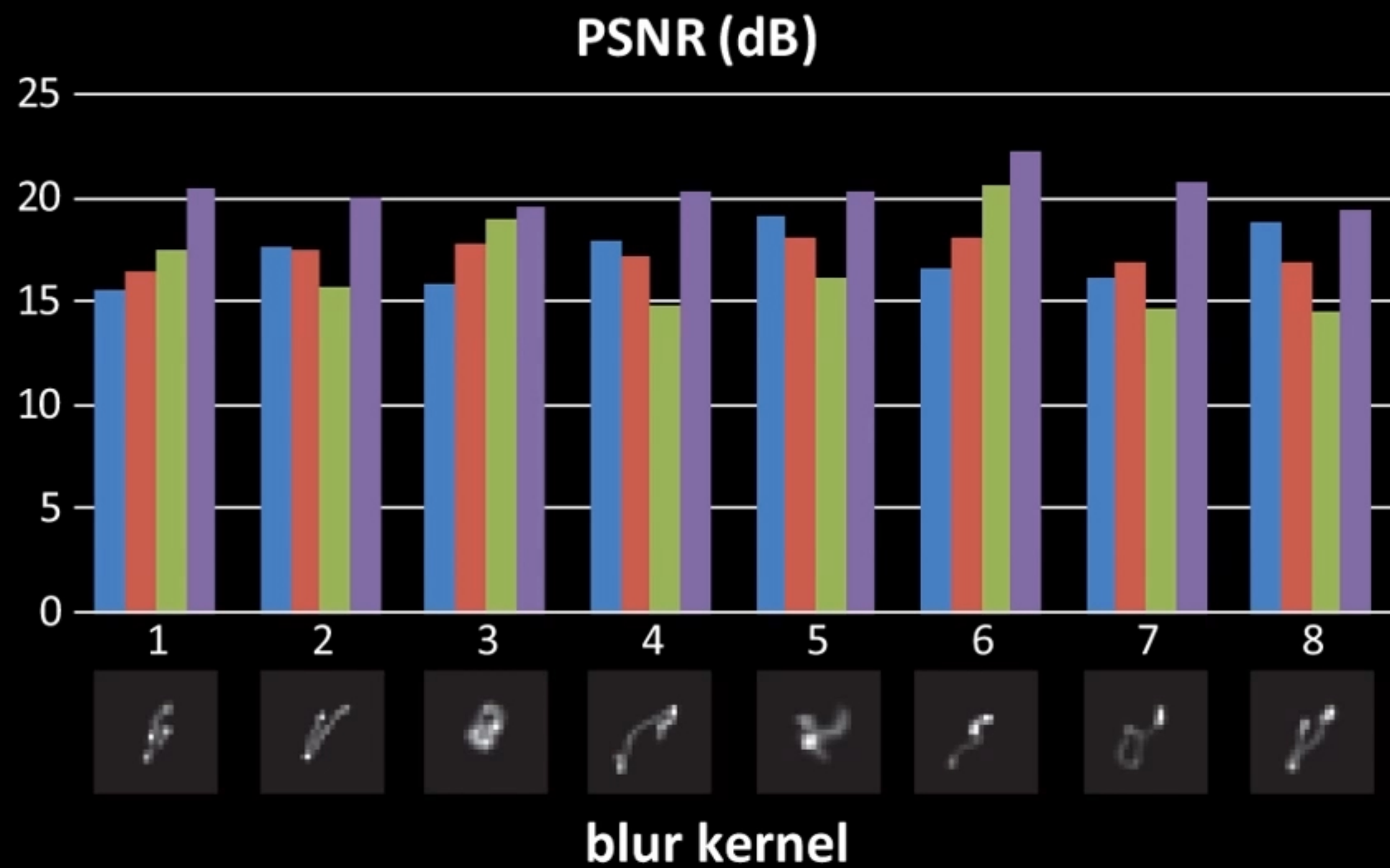
Figure 1. (a) the input image. (b) the corresponding feature space. The pixel A can be generated by linearly combining the color at B and C .

this method was extended by Chuang et al. [5] with the Bayesian estimation framework. These methods work well when the unknown pixels are near the foreground bound-

our result

Approach: predict blur kernel of figure region from surrounding text regions.

Quantitative evaluations



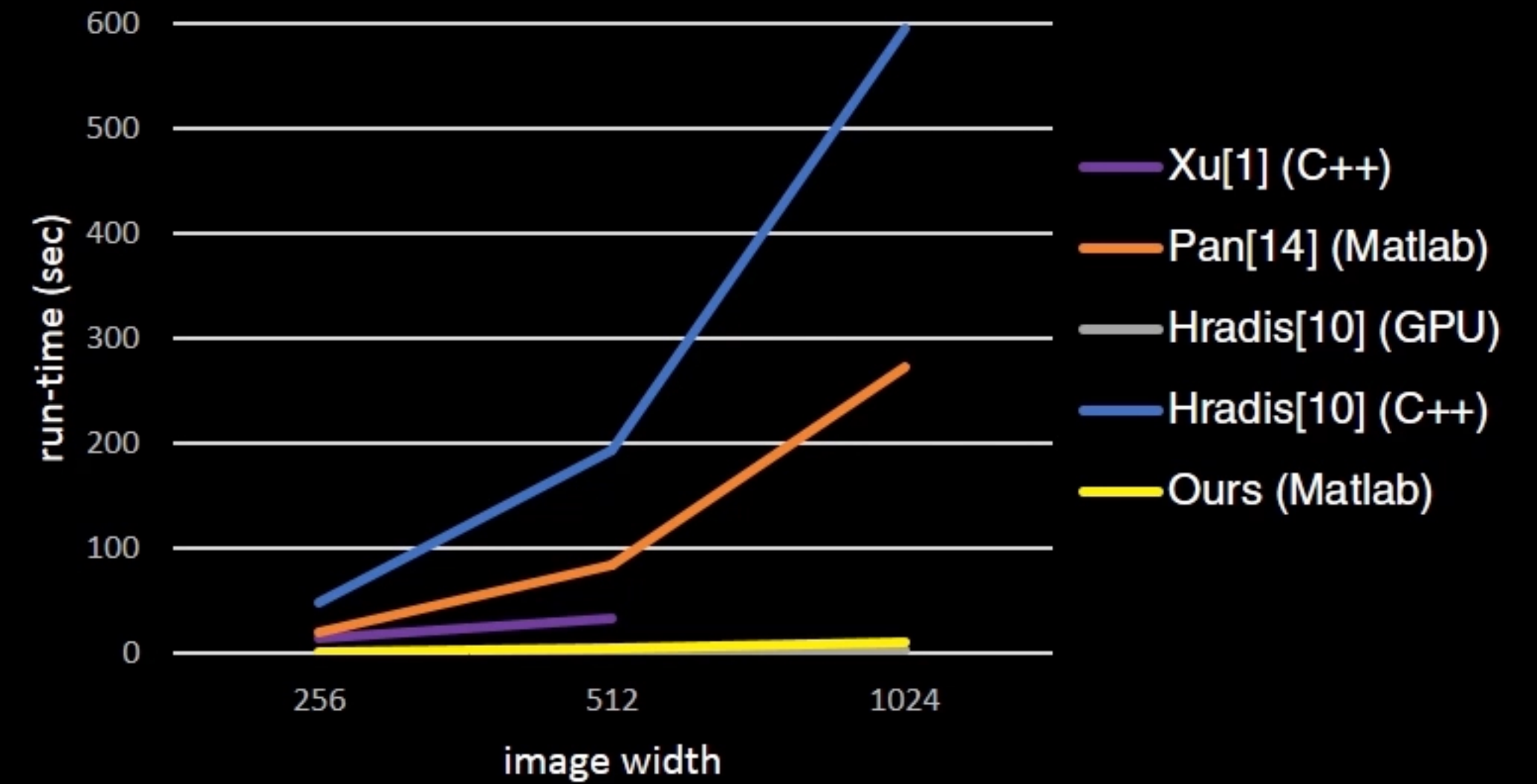
■ Xu ■ Pan ■ Hradis ■ Ours

Note the logarithmic scale markings in OCR results

High computational efficiency

Table 1: Run-time comparison (in seconds)

Image size	256 ²	512 ²	1024 ²
Xu[1] (C++)	14.8	33.4	-
Pan[14] (Matlab)	19.6	84.3	271.9
Hradiš[10] (C++)	48.5	193.7	594.9
Hradiš[10] (GPU)	0.3	1.0	3.1
Ours (Matlab)	2.0	3.9	11.4



Required operations: 2D FFT, 2D convolution, 1D Look-Up-Table.

Learning High-Order Filters for Efficient Blind Deconvolution of Document Photographs

Lei Xiao, Jue Wang, Wolfgang Heidrich, Michael Hirsch

UNIVERSITY OF BRITISH COLUMBIA

ADOBE RESEARCH

KAUST

MAX PLANCK INSTITUTE FOR INTELLIGENT SYSTEMS

Poster Session S-2B-08