Handling Noise in Single Image Deblurring using Directional Filter

Chris Andrew, Santhoshini Reddy, Ritu Srivastava

What is a blurred image?

- While taking a picture, camera shake is likely to happen and produce blurry pictures. This blur needs to be removed and we thus model the image and the blur in the form of a convolution operation wherein we then estimate a kernel that causes the blur.

How do we remove blurring?

In general we model a blurred image as follows:

$$b = (I * k) + n$$

where *I* is the unblurred sharp image, *k* is the blur kernel and *n* is added noise.

We then try and estimate the blur kernel and then do deconvolution to try and retrieve the unblurred image *I*.

How to estimate *k*?

There are a number of ways the blur kernel *k* is estimated, the most recent and state of the art method(Non deep learning based) proposed was:

- Remove noise from the blur image by using standard denoising operations.
- Estimate the blur kernel by optimising:

  $$k^* = \operatorname{argmin} \{ || b - (k^*I) || + P(k) \}$$

  where b is the blurred image, k is the blur kernel and I is the unblurred image. Both k and I are iteratively estimated until they are converged.

But there are issues with this algorithm, mainly that the denoising step hugely biases the kind of kernel that is estimated. The reason behind this is that essential edge information is lost/corrupted during denoising. This edge information is essential to kernel estimation.

How do we improve the algorithm?

Using directional filters to remove the noise can be used to remove the bias. This is because, when noise is removed by a directional filter, the direction orthogonal to the filter has the same radon transform projection as the original image. This property can be used to reconstruct the estimated kernel with the essential edge information.

$$R_{\theta'}(k_\theta) = R_{\theta'}(k * f_\theta) = R_{\theta'}(k) * R_{\theta'}(f_\theta) = R_\theta(k)$$

This is true when the difference in angles is 90 degrees.

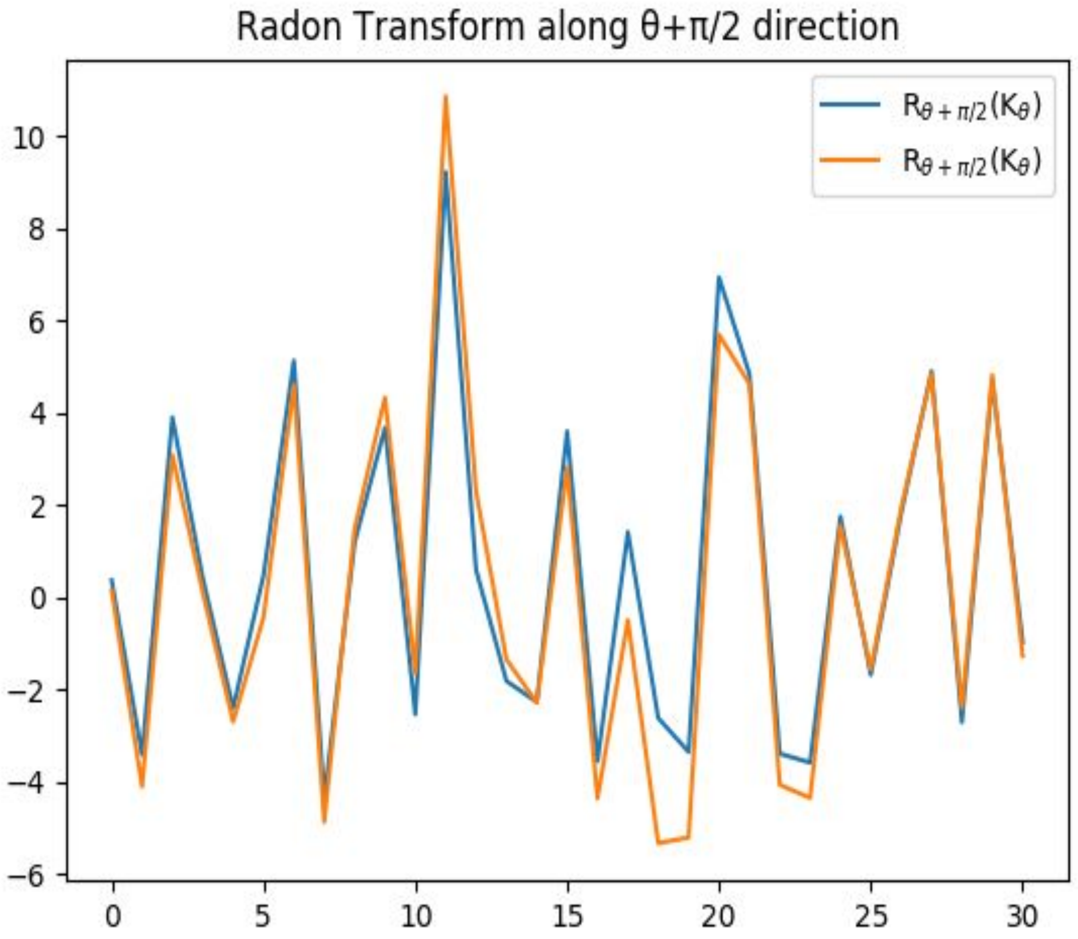
Effect of applying directional filter

Input image

Filtered image with 60° directional filter

Filtered image with 145° directional filter

Radon Transform



Proposed Algorithm

- Remove noise from the blur image by using directional filters.
- Compute the projection using Radon transform in the orthogonal direction
- Estimate the blur kernel by optimising:

  $$R(k^*) = R(\operatorname{argmin} \{ \| b - k^* I \| \})$$

  where b is the blurred image, k is the blur kernel and I is the unblurred image. Both k and I are iteratively estimated until they are converged.
- Reconstruct the actual *k* using inverse Radon transform.
- Deblur the image using Deconvolution with the estimated k.

Note**: This is not the actual algorithm, in essence this is what will be finally computed, the actual algorithm has been simplified.

Algorithm

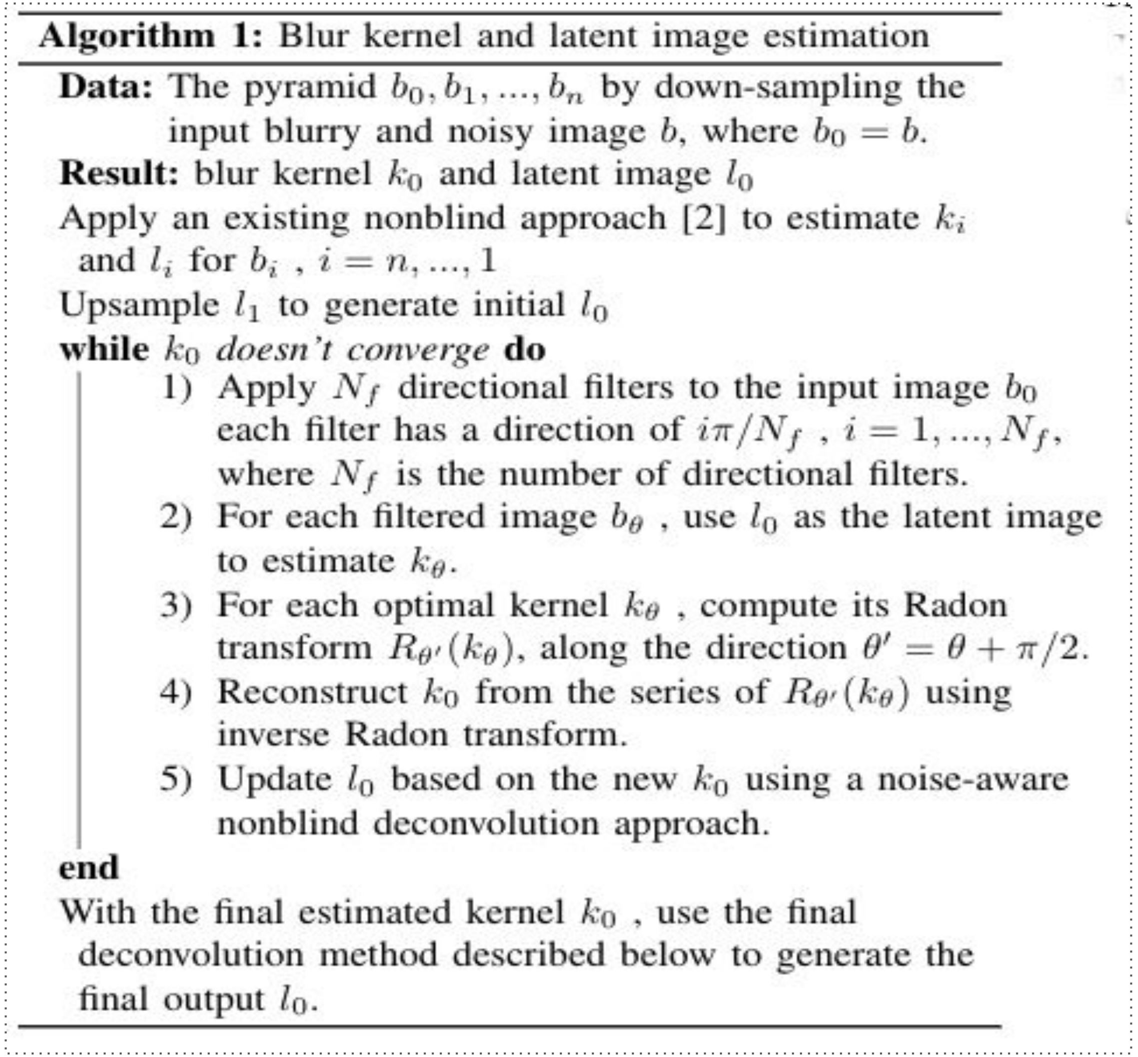
Initial estimation optimization:

$$L' = argmin_L ||b - (k * l)|| + \rho_L(L)$$

$$K' = argmin_L ||b - (k * l)|| + \rho_K(K)$$

Kθ estimation (Step 2):

$$k_\theta = argmin_{k_\theta}\{||\nabla b_\theta - k_\theta * \nabla l_0||^2 + \rho(k_\theta)\}$$



Algorithm 1: Blur kernel and latent image estimation

```
Data: The pyramid $b_0, b_1, ..., b_n$ by down-sampling the
      input blurry and noisy image $b$, where $b_0 = b$.
Result: blur kernel $k_0$ and latent image $l_0$
Apply an existing nonblind approach [2] to estimate $k_i$
  and $l_i$ for $b_i$ , $i = n, ..., 1$
Upsample $l_1$ to generate initial $l_0$
while $k_0$ doesn't converge do
    1) Apply $N_f$ directional filters to the input image $b_0$
       each filter has a direction of $i\pi/N_f$ , $i = 1, ..., N_f$,
       where $N_f$ is the number of directional filters.
    2) For each filtered image $b_\theta$ , use $l_0$ as the latent image
       to estimate $k_\theta$.
    3) For each optimal kernel $k_\theta$ , compute its Radon
       transform $R_{\theta'}(k_\theta)$, along the direction $\theta' = \theta + \pi/2$.
    4) Reconstruct $k_0$ from the series of $R_{\theta'}(k_\theta)$ using
       inverse Radon transform.
    5) Update $l_0$ based on the new $k_0$ using a noise-aware
       nonblind deconvolution approach.
end
With the final estimated kernel $k_0$ , use the final
  deconvolution method described below to generate the
  final output $l_0$.
```

Algorithm

L0 estimation (Step 5):

$$||\nabla l_0 * k_0 - \nabla b_0||^2 + w_1||\nabla l_0 - u(\nabla l_1)||^2 + w_2||\nabla l_0||^2$$

Here,

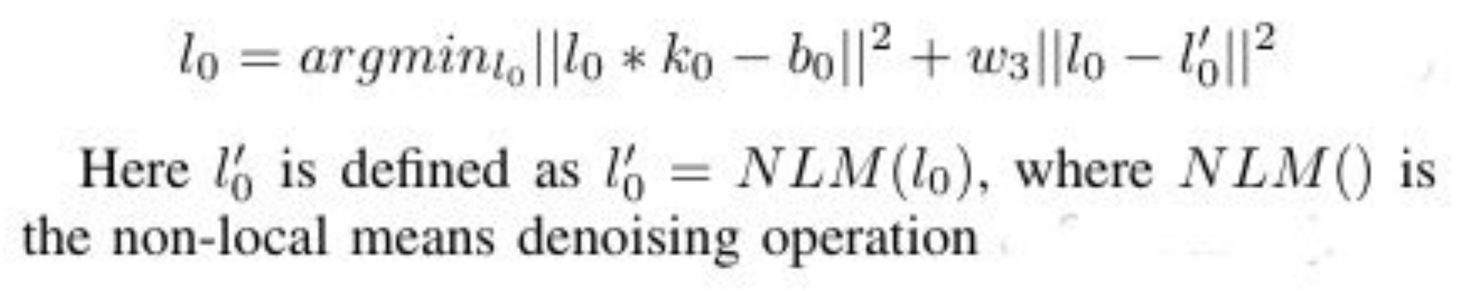
w1 and w2 are regularization constant,

u(.) is an upsampling function,

∇m represents the gradient of matrix m.

Latent Image Reconstruction

Final deconvolution step

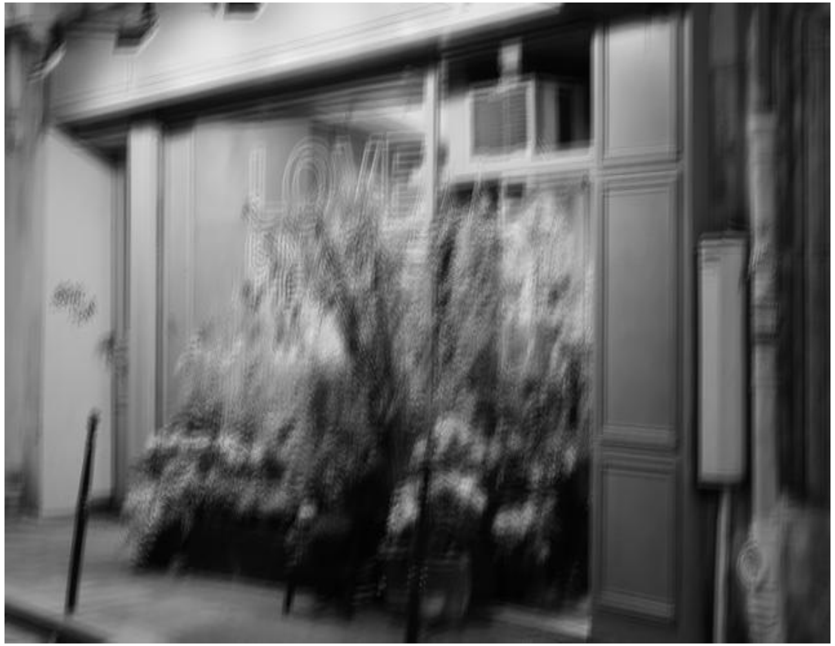


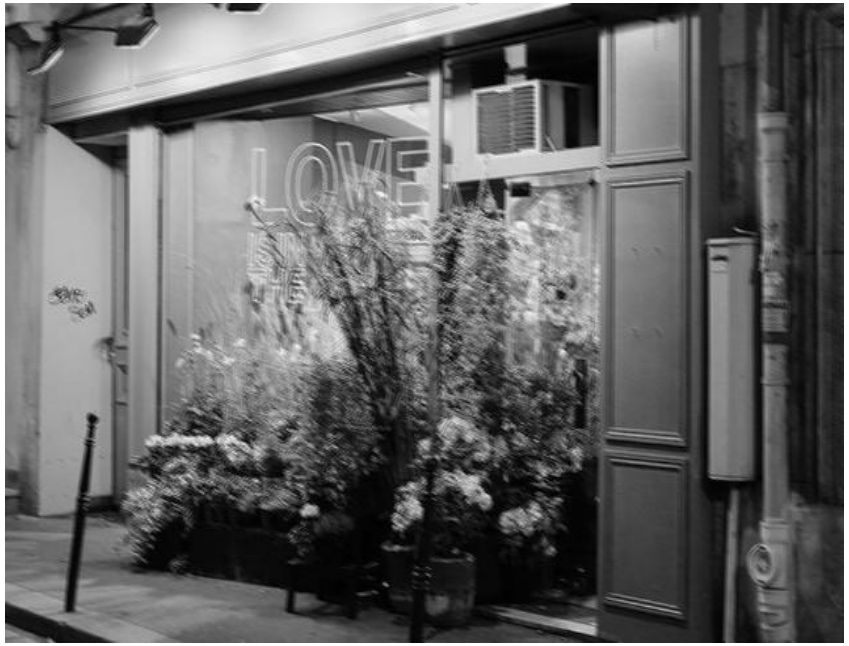
$$l_0 = argmin_{l_0}||l_0 * k_0 - b_0||^2 + w_3||l_0 - l_0'||^2$$

Here l_0' is defined as $l_0' = NLM(l_0)$, where $NLM()$ is the non-local means denoising operation

Blurred and Deblurred images

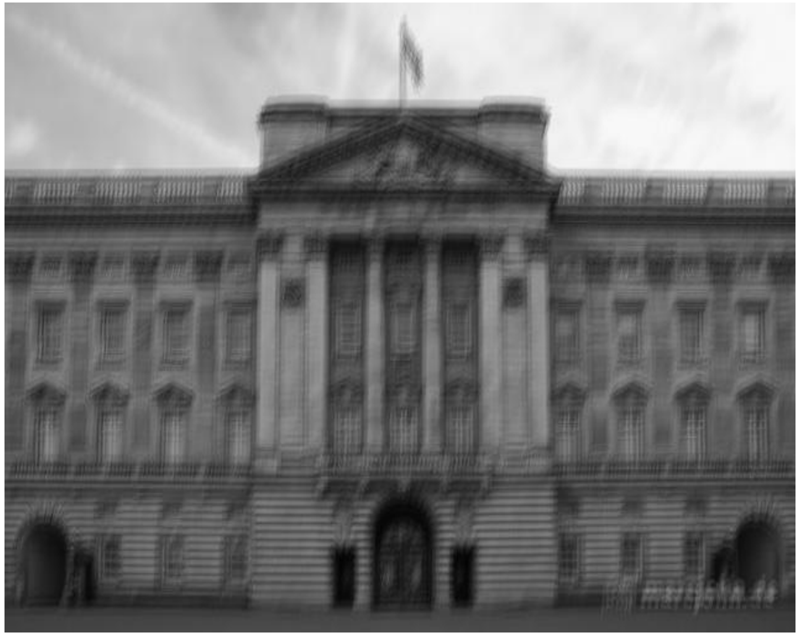
PSNR: 21.94

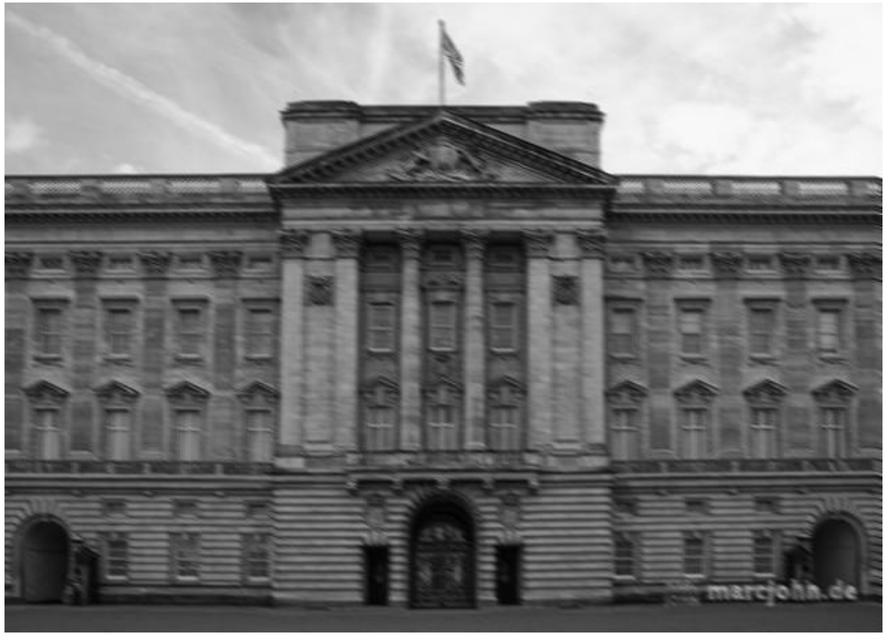




Blurred and Deblurred images

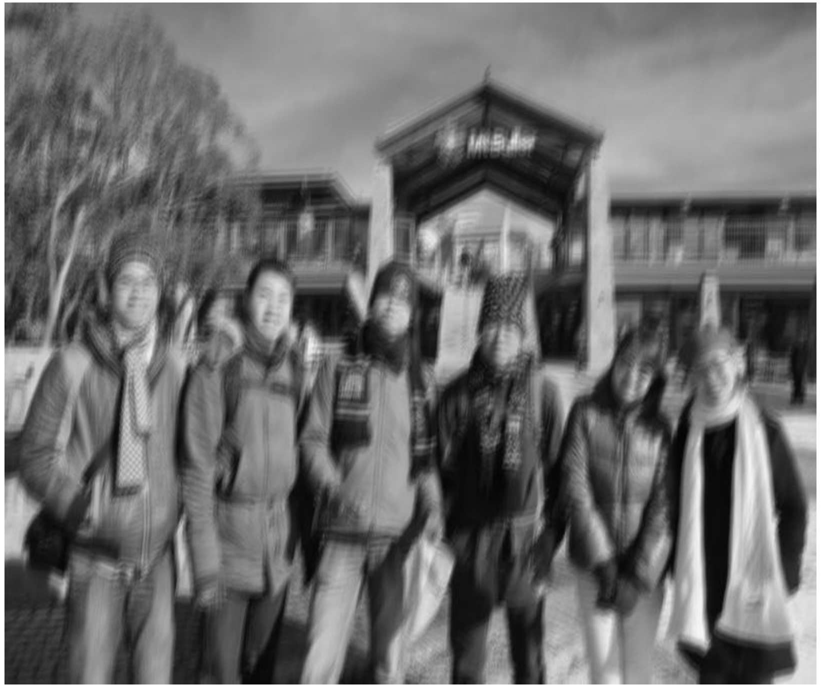
PSNR: 22.96

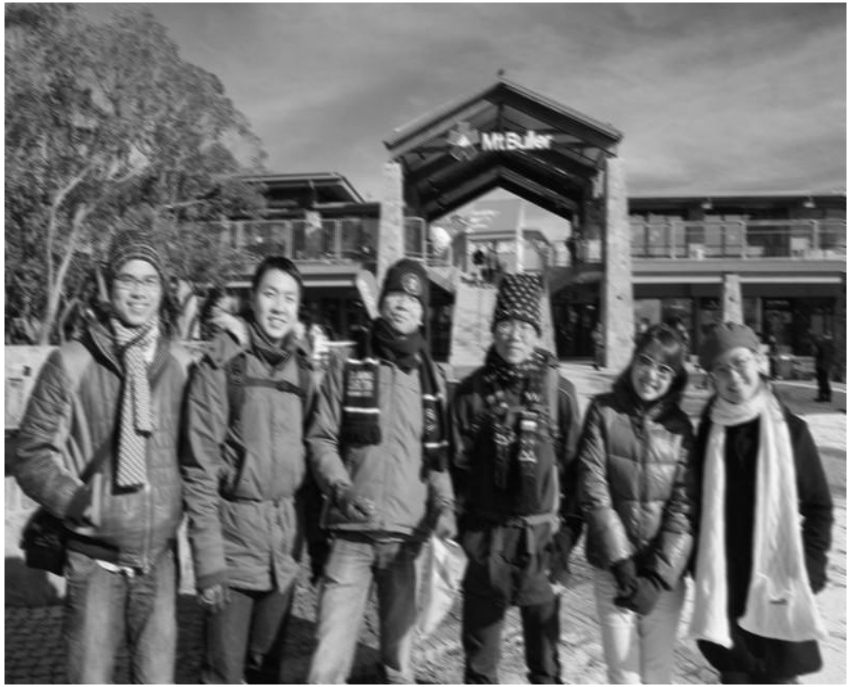




Blurred and Deblurred images

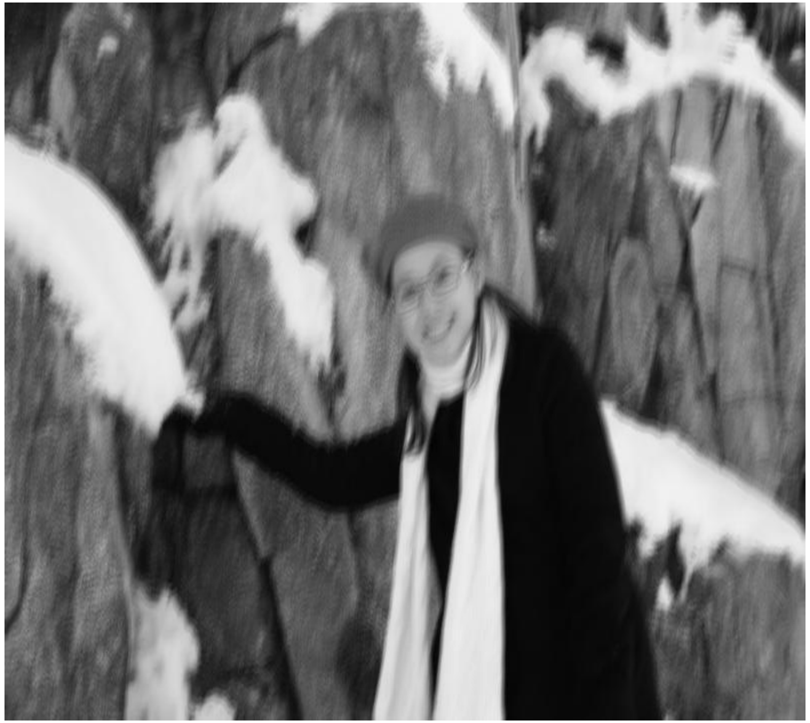
PSNR: 19.25

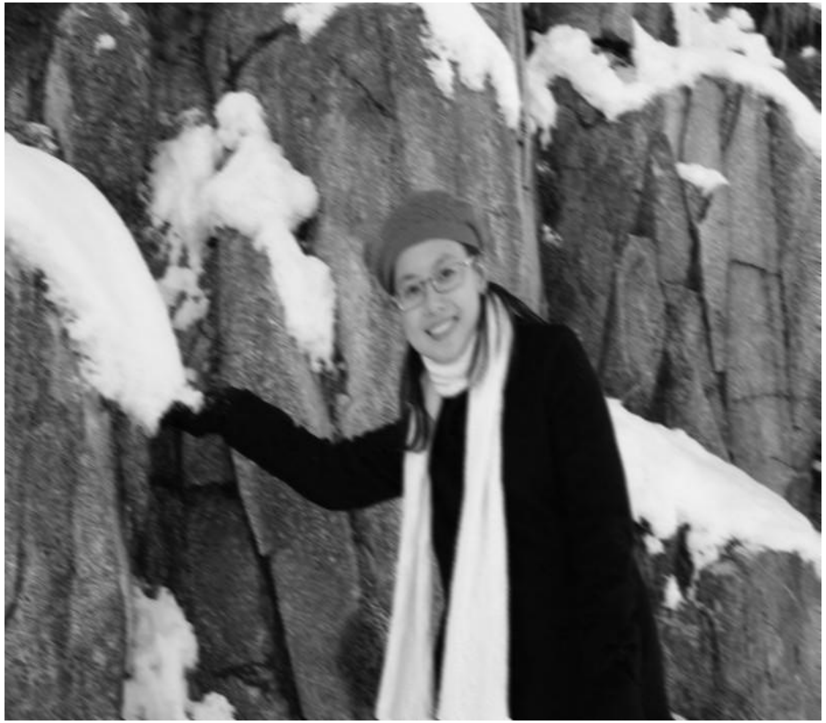




Blurred and Deblurred images

PSNR: 19.47





References

- Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. CVPR 2005.
- Sunghyun Cho and Seungyong Lee. Fast motion deblurring. ACM Transactions on graphics (TOG) 2009.
- Lin Zhong, Sunghyun Cho, Dimitris Metaxas, Sylvain Paris, and Jue Wang. Handling noise in single image deblurring using directional filters. CVPR 2013.
- Kernel Fusion dataset. Image blur dataset. http://web.cecs.pdx.edu/fliu/project/kernelfusion/