Handling Noise in Single Image Deblurring using Directional Filter

Chris Andrew, Santhoshini Gongidi, Ritu Srivastava

***Abstract*— This project is an implementation of the image deblurring algorithm proposed by Zhong *et. al* [4] in CVPR 2013. This project is done as part of the Computer Vision course at IIIT Hyderabad.**

I. Objective

While taking a picture, camera shake is likely to happen and this can produce blurry pictures. To restore the quality of the image, the camera blur and noise introduced in the captured image needs to be removed. To do this, we model the image and the blur in the form of a convolution operation using a blur kernel and addition of some random noise. We use this model to estimate the blur kernel and then perform deconvolution to get the de-blurred image.

II. Theory

In general, a blurred image is modeled as follows:

$$b = (l * k) + n \tag{1}$$

where l is the unblurred sharp image, k is the blur kernel and n is added noise. In this project, the goal is to estimate the blur kernel k and then do deconvolution to try and retrieve the unblurred image l or latent image. 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:

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

$$k* = argmin_k||b - (k * l)|| + P(k) \tag{2}$$

where b is the blurred image, k is the blur kernel and l is the unblurred image and P(k) is the regularization term. Both k and l 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.

We can prove that the denoising step biases the kernel by using Gaussian denoising to remove the noise. In the case of Gaussian, suppose we use a Gaussian kernel G_g to remove noise from the blurred image. We then try and find the blur kernel k_g as:

$$\begin{aligned} k_g &= argmin_{k_g}||b * G_g - l * k_g||^2 \\ &= argmin_{k_g}||(l * k + n) * G_g - l * k_g||^2 \\ &\approx argmin_{k_g}||l * (k * G_g - k_g)||^2 = k * G_g \end{aligned} \tag{3}$$

This shows that the kernel estimated k_g is biased from the actual blur kernel k. To overcome this bias, the proposed method uses Radon Transform to reconstruct the original k from the biased k_θ

III. Proposed Method

The method proposed in the paper follows a similar approach as mentioned earlier: remove noise by denoising the image, estimate the blur kernel, deconvolve to get latent image.

We know that the kernel estimation is biased because of the denoising step, therefore the paper proposes using directional filtering as the denoising step to prevent the kernel from being biased.

Directional filtering is defined as:

$$I(p) * f_\theta = \frac{1}{c}\int_{-\infty}^{\infty} w(t) I(p + tu_\theta)dt \tag{4}$$

Where $w(t)$ is the Gaussian kernel, defined by $w(t) = e^{-t^2/2\sigma_f^2}$. $I(p)$ is the intensity value at point p and $u_\theta = [sin\theta, cos\theta]^T$ is the direction vector.
Since we use a Gaussian kernel, filtering this image as $b_\theta = b * f_\theta$, this gives us the blur kernel as:

$$k_\theta = k * f_\theta \tag{5}$$

We now want to retrieve the actual blur kernel k from k_θ, to do this we use the Radon transform. The Radon transform has the property that a projection orthogonal to the direction of the image, will have almost no effect on the projection. Therefore, we project the estimated blur kernel k_θ at $\theta' = \theta + \frac{\pi}{2}$. The projection is formulated as:

$$R_{\theta'}(k_\theta) = R_{\theta'}(k * f_\theta) = R_{\theta'}(k) * R_{\theta'}(f_\theta) = R_{\theta'}(k) \tag{6}$$

We can then compute the inverse Radon transform to get the actual blur kernel k.

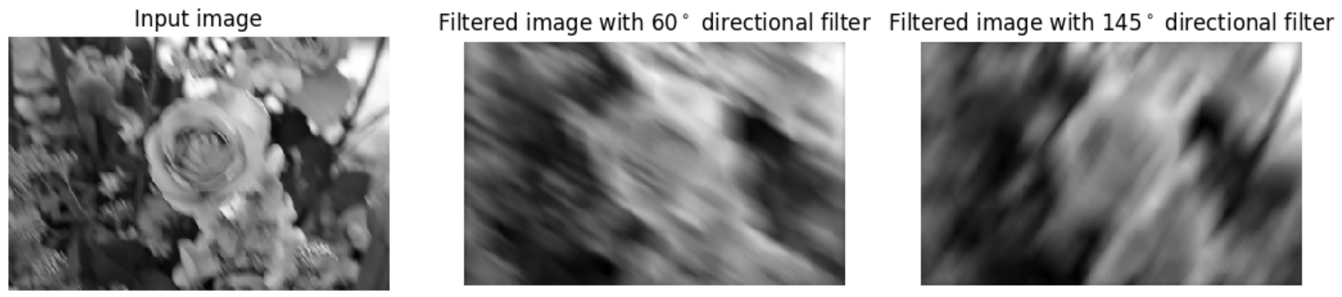
Input image
Filtered image with 60° directional filter
Filtered image with 145° directional filter

Fig. 1: Directional filter results for the input image at 60° and 145° respectively

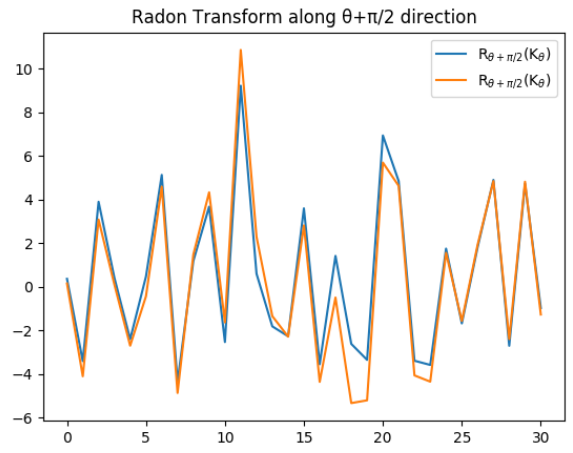

Fig. 2: Results showing that radon transform of blur kernel K is preserved along orthogonal direction.

The entire algorithm is given as:

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 $l_0$.
```

For estimating k_θ in step (2), we use the following optimisation:

$$k_\theta = argmin_{k_\theta}\{||\nabla b_\theta - k_\theta * \nabla l_0||^2 + \rho(k_\theta)\} \quad (7)$$

Once we have an estimate for k_0, we can estimate l_0 by minimising the following energy function:

$$||\nabla l_0 * k_0 - \nabla b_0||^2 + w_1||\nabla l_0 - u(\nabla l_1)||^2 + w_2||\nabla l_0||^2 \quad (8)$$

Where w_1 and w_2 are regularisation constants and $u(.)$ is the upsampling function.

The final deconvolution step mentioned in the end to obtain the final l_0 is defined as optimising the following function:

$$l_0 = argmin_{l_0}||l_0 * k_0 - b_0||^2 + w_3||l_0 - l_0'||^2 \quad (9)$$

Here l_0' is defined as $l_0' = NLM(l_0)$, where $NLM()$ is the non-local means denoising operation defined in [1].

IV. WORK DONE AND IMPLEMENTATION DETAIL

This project was implemented in Python programming language using scikit-image package for image processing, NumPy package for fast and efficient array calculations and PyTorch for optimization using gradient descent optimization. The following modules were implemented in this project.

- Implemented module to compute initial estimate of blur kernel and latent image. The reference work uses S.Cho's[2] fast deblurring algorithm to get initial estimates. In this module, we optimize the following equations iteratively.

$$L' = argmin_L||b - (k * l)|| + \rho_L(L) \quad (10)$$

$$K' = argmin_L||b - (k * l)|| + \rho_K(K) \quad (11)$$

The functions $\rho_L(L)$ and $\rho_K(K)$ act as the regularization terms in the above equations. Different functions are used in the reference work. In our work, we use L2 norm as the regularization factor. The optimal values L' and K' for the above objective functions are learnt using gradient descent in our implementation.

- Implemented module to get the directional filtered images along different directions.

- Implemented module to estimate the biased kernel k_θ for a given directional filtered image b_θ.
- Implemented module to calculate Radon Transform the biased at an orthogonal angle to the direction of the filter.
- Implemented module to estimate the unbiased kernel using inverse radon transform for a given series of radon projections.
- Implemented module to use an unbiased estimation of k to find an estimate of the latent image.
- Implemented module to perform the final deconvolution to estimate the final latent image using an optimised estimation of k and non-local means denoising method.

TABLE I: Blurred input images (Best viewed when zoomed)

V. Experiments and Dataset

To test the image blurring algorithm proposed in the reference work, we will run the algorithm on both real images that have natural blur and noise and synthetically generated images. The test data is sampled from the dataset available at [3]. Table I shows examples for different blurred images.

The figures in Table III shows the deblurred results obtained for different input images using our implementation, we also show one of the blur kernels estimated by our method in Fig. 3.

We also calculate the PSNR for the blur image and the final obtained latent image using the following formulation:

$$PSNR = 10\log_{10}(\frac{R^2}{MSE}) \tag{12}$$

where R is the maximum fluctuation of the values possible(maximum possible intensity value - minimum possible intensity value) and MSE is the Mean Squared Error given by:

$$MSE = \frac{\sum_{M,N}[I_1(i,j) - I_2(i,j]^2}{M * N} \tag{13}$$

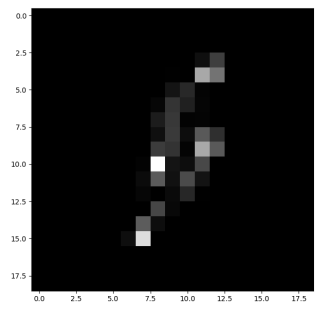

Fig. 3: One of the estimated kernels using the method described and our implementation

We calculated the PSNR values for our deblurred and the blurred images and the results for them are shown in the Table II. These results are very similar to the ones obtained in the paper which were computed for three different images which are also given in the same table.

Own Image	**PSNR**	**Original Paper**	**PSNR**
Image 1	21.94	Abbey	22.73
Image 2	22.26	Chalet	22.80
Image 3	19.25	Aque	28.46
Image 4	19.47	-	-

TABLE II: PSNR values calculated for the 4 images that we deblurred using our implementation of the paper and the PSNR values calculated for 3 different images as reported in the paper

VI. GitHub Link

We used GitHub for project, all the code and it's development was done incrementally and is available on the GitHub repository. The repository can be accessed on:

https://github.com/chrizandr/vision-project

References

[1] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[2] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. *ACM Transactions on graphics (TOG)*, 28(5):145, 2009.

[3] Kernel Fusion dataset. Image blur dataset. http://web.cecs.pdx.edu/~fliu/project/kernelfusion/.

[4] Lin Zhong, Sunghyun Cho, Dimitris Metaxas, Sylvain Paris, and Jue Wang. Handling noise in single image deblurring using directional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 612–619, 2013.

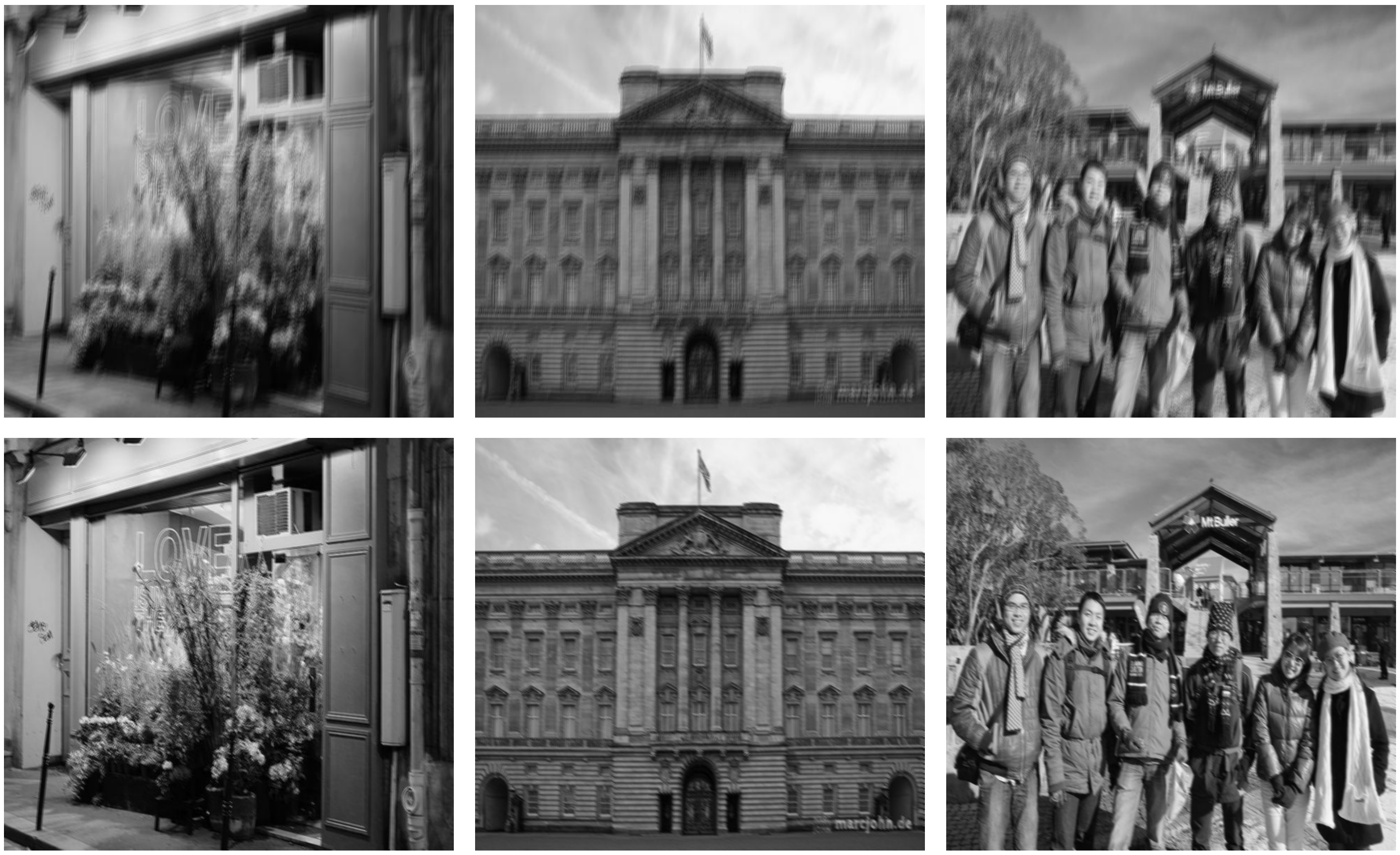

TABLE III: Results obtained for blurred images. First row displays the input images and second row shows the deblurred output. (Best viewed when zoomed)