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:

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

  where b is the blurred image, k is the blur kernel and l is the unblurred image. 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.

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.

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:

  $$R(k^*) = R(\text{argmin } \{ \| b - k*l \| \})$$

  where b is the blurred image, k is the blur kernel and l is the unblurred image. Both k and l 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.

Progress of the project - I

- Module to get initial estimate of blur kernel and latent image is completed. The reference work uses S.Cho's work to get initial estimates. In this module, we optimize the following equations iteratively.

$$
\begin{aligned}
L' &= \operatorname{argmin}_L \{\|B - K * L\| + \rho_L(L)\}, \\
K' &= \operatorname{argmin}_K \{\|B - K * L\| + \rho_K(K)\}.
\end{aligned}
$$

This optimization is solved using gradient descent. B is blur image, K and L are the blur kernel and latent image respectively. $\rho_L(L)$ and $\rho_K(K)$ functions act as regularization terms. In this optimization, these functions calculate the total variation of the matrix.

Progress of the project - II

- Implemented module to compute initial estimate of blur kernel and latent image. The reference work uses S.Cho’s fast deblurring algorithm to get initial estimates.
- mplemented module to get the directional filtered images along different directions.
- Implemented module to calculate radon transform of a image at a different angles.
- Implemented module to calculate an inverse radon transform for a given series of radon transforms.

Work to be done

- Module to estimate a latent image for a given B and K using noise aware deconvolution. This step also involves optimizing two objective functions iteratively.
- Module to estimate K along a direction θ using a blur image B and latent image estimate L
- Experiment the performance of this algorithm for different blurred images.
- Will try to compare these results with deep learning based deblurring algorithm results.

Dataset

Real data and synthetic data has been used in this work.

Available datasets that we can use in our project:

- https://riemenschneider.hayko.at/vision/dataset/task.php?did=382
  - 2976 images with GT and kernel estimations
  - Gray scale images
- https://mklab.iti.gr/results/certh-image-blur-dataset/
  - 2450 images with GT
  - RGB images.