Linear Motion Deblurring

Kristian Ryder Thomsen and Kristian Berg Thomsen
02625 CSI: Computational Science in Imaging, DTU Compute

Blurring the image

Image blurring is assumed to be a linear operation. Denote by x and b the vectorized sharp and blurred image respectively. The image consists of $N = m \times n$ pixels. Because of linearity there must exist a (huge) matrix $A \in \mathbb{R}^{N \times N}$ such that x and b are related by the fundamental linear model of image blurring $Ax = b$.
The naive solution $x_{naive} = A^{-1}b$ fails due to effects of inverted noise. In special cases with specific boundary conditions (BC) and point spread function (PSF) fast non-iterative algorithms exist, but otherwise iterative methods are needed.

Linear motion blur is motion blur along a straight line. If linear motion blur is not parallel with one of the image axes this gives rise to a PSF that is neither separable nor doubly symmetric. This means that we cannot use periodic BC and hence we must use iterative methods.

Linear motion blur is characterized by its angle α and length L. The angle is measured in degrees and the length in pixels. The angle decribes the motion direction w.r.t horizontal. The length decribes how much the camera/photographed object was moved during the exposure. Examples of blurred images and the corresponding parameters α and L can be seen in the bottom of the second and third column.

For $\alpha = 10$ and $L = 15$ an explictly formed A matrix is seen to the left. The A matrix is normally never explictly formed and it is formed here only for illustrative purposes. Note that the sum of each row is approximately constant. This will impact the iterative methods presented later.

Advantages of iterative methods

Using only matrix-vector multiplications iterative methods produce a sequence $x^{[0]} \rightarrow x^{[1]} \rightarrow x^{[2]} \rightarrow \ldots$ that (hopefully) converge to the desired solution. This gives two major advantages

- The matrix A is never altered, and is only "touched" when computing Ax and $A^T y$.
- The matrix A is not explictly required - we only need a "black box" that computes the action of A or its underlying operator.

Computing the multiplication Ax corresponds to a blurring of the image stored vectorized in x. This means that Ax and $A^T x$ can be computed by a filter operation without forming A (nor A^T) explictily.

The image deblurring problem is an inverse problem.
This is difficult for several reasons

- The matrix A is huge.
- The matrix A is very ill-conditioned.
- The matrix A may be an imprecise model of the blurring.

Iterative methods

Since the linear motion blur does not necessarily give rise to a PSF which is separable or doubly symmetric we have to use iterative methods. We consider three such methods, namely, Landweber's method, Cimmino's method, and CGLS (conjugate gradients for least squares). All these algoritms exhibit semiconvergence, that is, initial convergence towards x_{exact} followed by slow convergence to $A^{\dagger}b$. We can obtain an approximate solution by choosing the number of iterations correctly.

Landweber's method

$x^{(k+1)} = x^{(k)} + \omega A^{\top}\left(b - Ax^{(k)}\right)$
where ω is a parameter chosen such that $0 < \omega < 2||A^{\top}A||_2^{-1}$. This method is a simple version of Cimmino's method, if the weights d_i are almost constant.

Cimmino's method

$x^{(k+1)} = x^{(k)} + \omega A^{\top} D\left(b - Ax^{(k)}\right)$
where D is a diagonal matrix with weights determined by the rows a_i of A. The weights are given by $d_i = \frac{1}{m||a_i||_2^2}$ when $||a_i||_2 \neq 0$. Otherwise $d_i = 0$.

Conjugate gradients for least squares

To motivate the CGLS method we first need to introduce the *Krylov subspaces*

$$\mathcal{K}_k = \text{span}\left\{A^{\top}b,\ A^{\top}AA^{\top}b,\ \left(A^{\top}A\right)^2 A^{\top}b,\ \ldots,\ \left(A^{\top}A\right)^{k-1} A^{\top}b\right\}.$$

We now search for a solution in this space. In each iteration we solve

$$x^{(k)} = \text{argmin}_x ||Ax - b||_2, \quad x \in \mathcal{K}_k,$$

from which one can derive a simple iteration scheme.

Note that in our implementation of the above methods we never need to form the matrix A explicitly.

Results

We can now present the first results using the CGLS algorithms with 40 iterations. All images are quadratic and the length of the blur is given in percent of the image width. Similar results are obtained with Landweber's method.

Fig. 1: $\alpha = 0$, $L = 6\%$

Fig. 2: $\alpha = 30$, $L = 7\%$

Fig. 3: $\alpha = 120$, $L = 14\%$

Fig. 4: $\alpha = 170$, $L = 18\%$

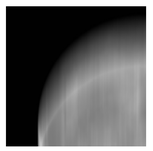
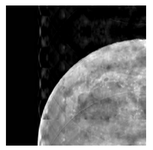


Fig. 5: $\alpha = 90$, $L = 37\%$



Estimating parameters

To obtain the results before we had to know how the images had been blurred, i.e. we had to know the angle and the length of the blurring. Now we want to deblur an image without "a priori" knowledge of angle and length. For this we will use the cepstral method. The cepstrum $\mathcal{C}(f)$ is defined as

$$\mathcal{C}(f) = \mathcal{F}^{-1}(\log|\mathcal{F}(f(x, y))|).$$

Below we see a blurred image, the Fourier transform and the cepstrum of the image. To reduce boundary effects in the cepstrum we apply a Hann window to the blurred image.

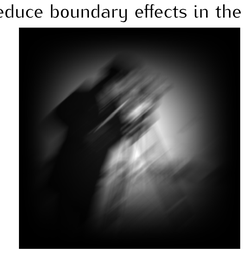


Fig. 11: Blurred image with Hann window

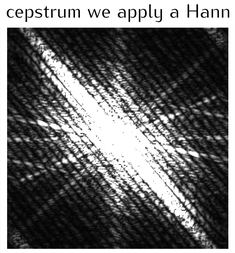


Fig. 12: Fourier transform

Fig. 13: Cepstrum

To estimate the angle we have to find the slope of the line connecting the two smallest points in the cepstrum. We know that

$$g(x, y) = f(x, y) * h(x, y) \quad \rightarrow \quad \hat{g}(x, y) = \hat{f}(x, y) \cdot \hat{h}(x, y),$$

and that $\hat{h}$ is the sinc function. We want to detect the ripples in the Fourier spectrum. Finding the two points in the third plot above we can calculate the angle as the slope of line though the points. The length of the blur is calculated as half the distance between the two points due to symmetry in the Fourier spectrum.

Results

For images without noise the above method is very effective. We have also plotted the error of the estimates for constant length and constant angle, respectively.

Fig. 14: Blurred image, $\alpha = 45$ $L = 6\%$

Fig. 15: Deblurred im., $\alpha_{\text{est}} = 45.07$ $L_{\text{est}} = 6.09\%$

Fig. 16: Blurred image, $\alpha = 22$ $L = 18\%$

Fig. 17: Deblurred im., $\alpha_{\text{est}} = 21.80$ $L_{\text{est}} = 17.96\%$

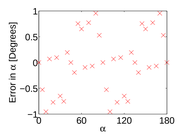


Fig. 18: Error for angle estimation, $L = 10\%$

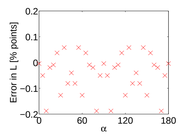


Fig. 19: Error for length estimation, $L = 10\%$

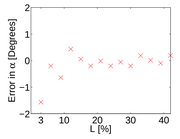


Fig. 20: Error for angle estimation, $\alpha = 25$

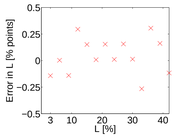


Fig. 21: Error for length estimation, $\alpha = 25$