Total Variational Blind Deconvolution

By Nikhil Agarwal and Abhijeet Parida

Supervised by Thomas Moellenhoff

nikhil.agarwal@tum.de abhijeet.parida@tum.de

The Task

Given a BLURRY image

Without a blur kernel

Compute DEBLURRED image

By estimation of Blur Kernel

Convoluted Image

$$f = k_0 * u_0 + n$$

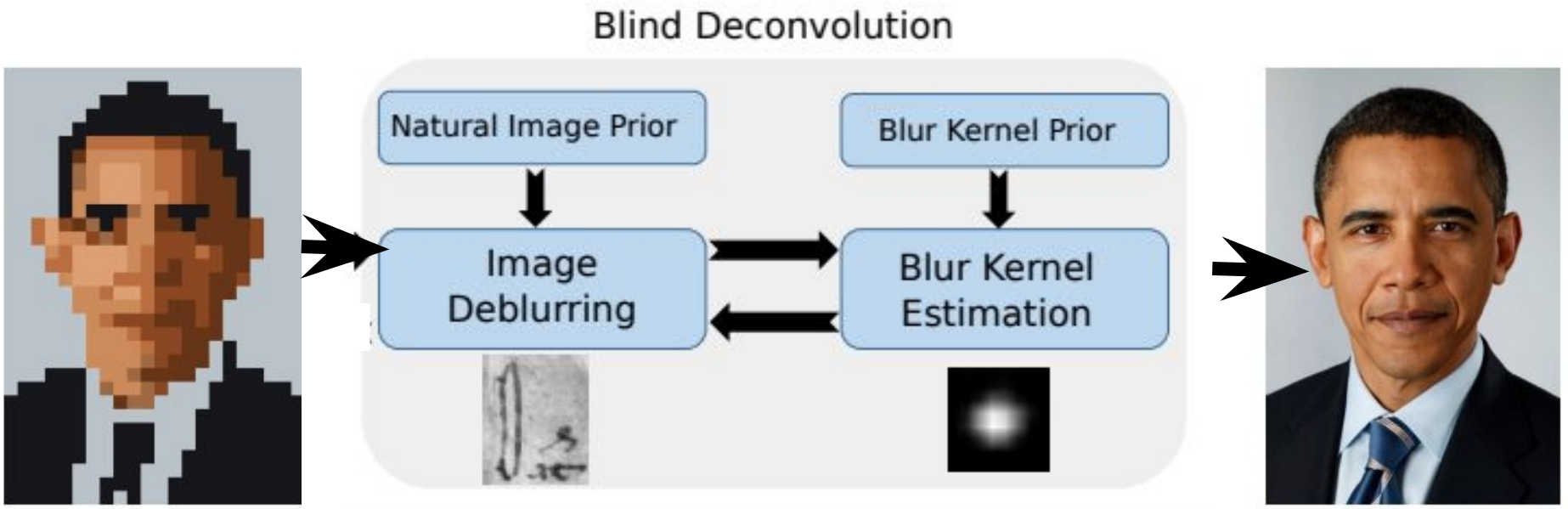
where,

F is blurry image
k_0 is a blur kernel,
u_0 is a sharp image,
n is noise and
$k_0 * u_0$ denotes convolution between k_0 and u_0 .

Blind Deconvolution

- Deconvolution is the process of restoring original image from the convoluted result
- It is an energy minimisation problem
- If the convolution kernel is unknown it is called Blind Deconvolution

Blind Deconvolution



Source- Koehler T; Binarisation Driven Blind Deconvolution

Regularised Energy & Minimization

$$\begin{aligned} &\min_{u,k} && ||k * u - f||_2^2 + \lambda J(u) \\ &\text{subject to} && k \succcurlyeq 0, \quad \|k\|_1 = 1 \end{aligned}$$

where

the first term enforces the convolutional blur model

J(u) is smoothness priors for u, and

λ is nonnegative weight.

Modelling of Smoothness Prior, J

Using Total Variation (TV) Norms - L1 norms of the derivatives

$$J(u) = ||u_x||_1 + ||u_y||_1$$

the gradient of u

$$\nabla u \doteq [u_x \; u_y]^T$$

Source - Rudin et al

Alternating Minimisation

- Minimisation of image energy using a guessed or updated kernel

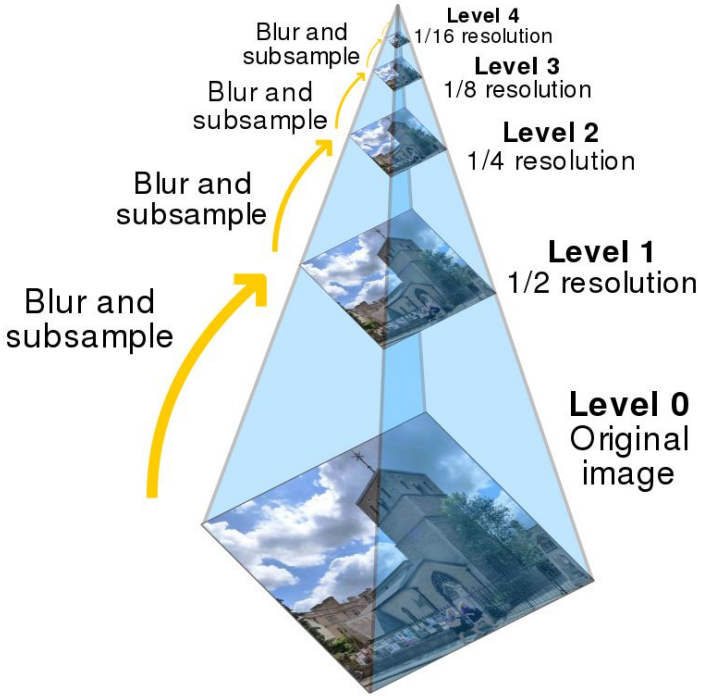
$$u^{t+1} \leftarrow \min_u ||k^t * u - f||_2^2 + \lambda J(u)$$

- Minimisation of kernel energy using the energy minimised image

$$\begin{aligned} k^{t+1} \leftarrow \quad & \min_k \quad && ||k * u^t - f||_2^2 \\ & \text{subject to} && k \succcurlyeq 0, \quad \|k\|_1 = 1. \end{aligned}$$

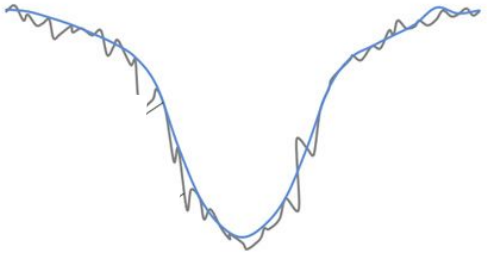
Implementation

Image Scaling - Pyramid Scheme

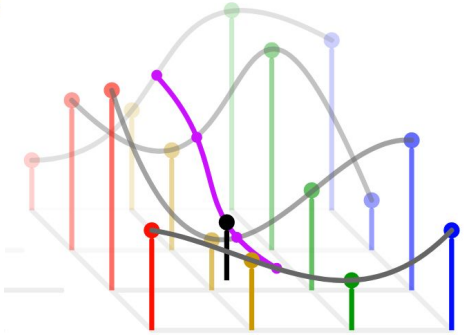


Source - Wikipedia(Pyramid_Scheme)

Why Blur further?



How it is Blurred ?



Bicubic Interpolation

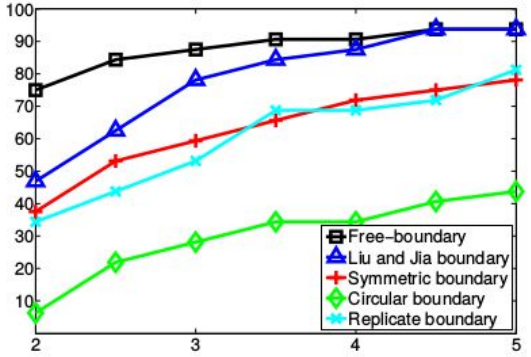
Source - Wikipedia(Bicubic_Interpolation)

Algorithm

Data: f, size of blur, initial large λ, final λ_{min}
Result: u,k

$u_0 \leftarrow \text{pad}(f)$;
$k_0 \leftarrow \text{uniform}$;
while *not converged* **do**
  $u^{t+1} \leftarrow u^t - \epsilon_u \left(k^t_{\_} \bullet (k^t \circ u^t - f) - \lambda \nabla \cdot \frac{\nabla u^t}{|\nabla u^t|} \right)$;
  $k^{t+1/3} \leftarrow k^t - \epsilon_k \left(u^{t+1}_{\_} \circ (k^t \circ u^{t+1} - f) \right)$;
  $k^{t+2/3} \leftarrow \max\{k^{t+1/3}, 0\}$;
  $k^{t+1} \leftarrow \frac{k^{t+2/3}}{\|k^{t+2/3}\|_1}$;
  $\lambda \leftarrow \max\{0.99\lambda, \lambda_{min}\}$;
  $t \leftarrow t + 1$;
end
$u \leftarrow u^{t+1}$;
$k \leftarrow k^{t+1}$;

cumulative histogram of deconvolution error ratio



Selection of Convoluted Region

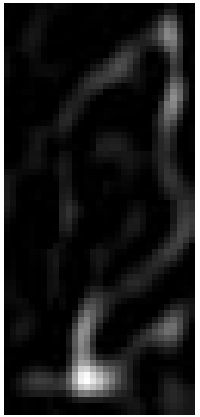
Source- Perrone, Favarro, Total Variation Blind Deconvolution

Result

Result

Blurry Input

Deblurred Output



Est. Kernel*

*kernel is not to scale


Toronto

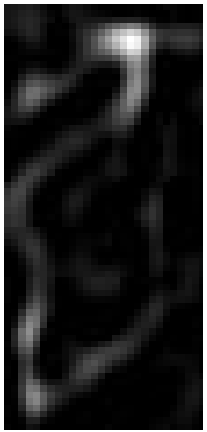
Images: Comparison with MATLAB

MATLAB Output

CUDA Output

Kernel: Comparison with MATLAB

MATLAB Output

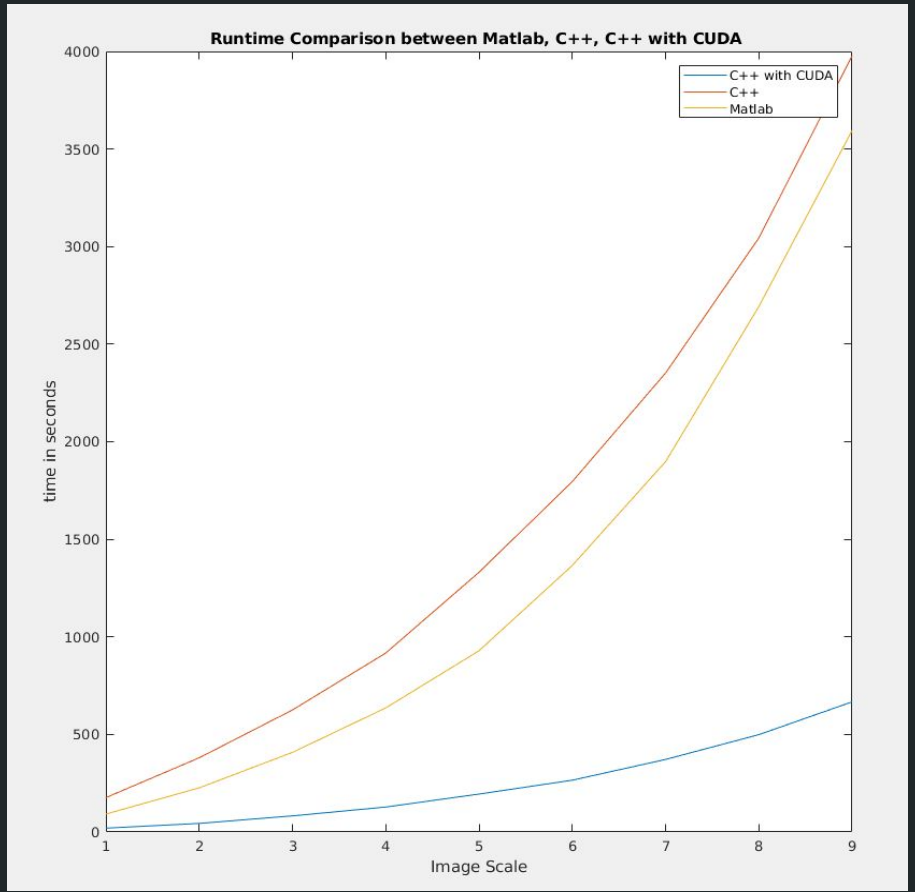


CUDA Output

*kernel is not to scale

Runtime Comparison*

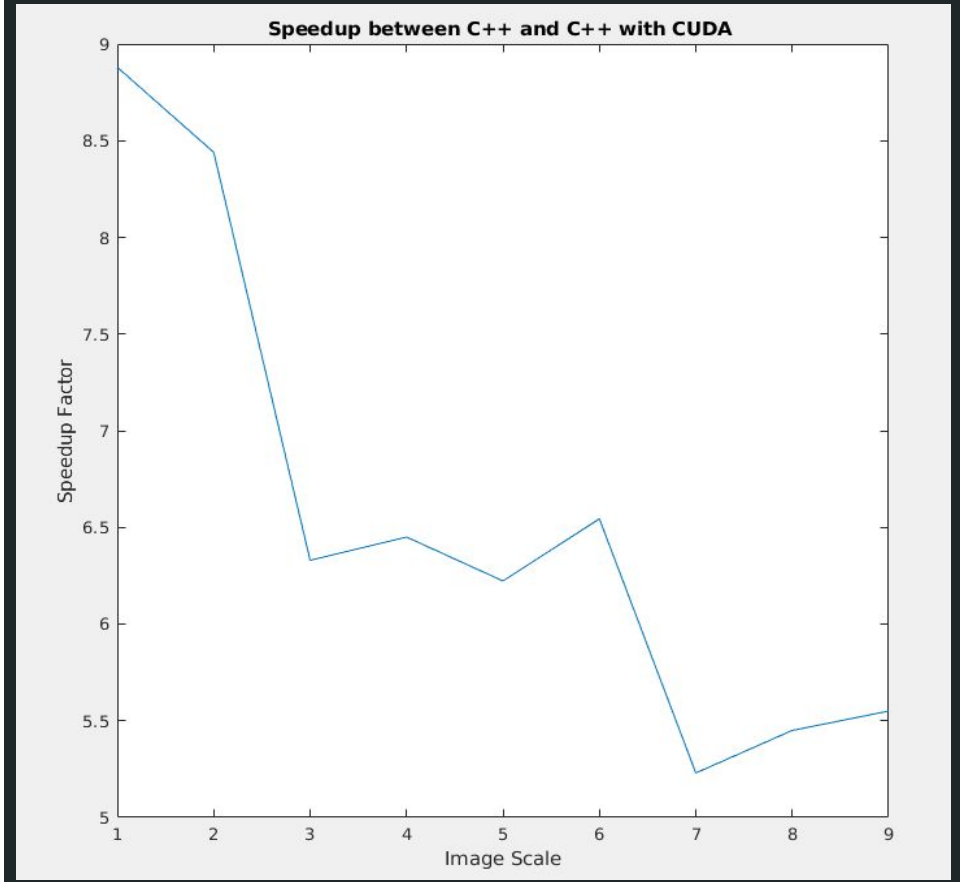
1. MATLAB,
2. C++ and
3. C++ with CUDA



* The results displayed may vary from system to system

CUDA SpeedUp*

C++ with CUDA over C++



The results displayed may vary from system to system

Challenges Overcomed

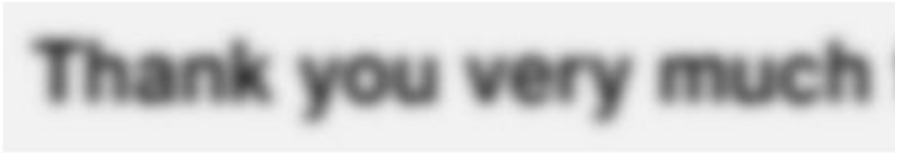
1. OpenCV assignment is by reference ➜ clone
2. Using std::Vector Library with OpenCV
3. Management of large C++ project

What is done?

1. CUDA - Total Variation
2. CUDA - Naive Convolution
3. Cublas Mathematical Operation

What is pending?

1. CUDA - FFT Convolution
2. CUDA - Interpolation (Texture Memory)
3. Kernel Stream and Async Copy
4. GPU enabled OpenCV


Thank you very much


Thank you very much


Thank you very much