Introduction to Project Topics

Kaicong Sun

Organizational Issues

- Projects can be done in groups of two or three.
- Submit the codes and report to Kaicong.Sun@ipvs.uni-stuttgart.de
  - Template in ILias
  - Workflow
  - Codes with comments
  - If possible, compare with CPU implemetation
- Written report: 6-12 pages
- Submission deadline: 31.03.2019

Topics

- Implementation of the Norm for Modulation Transfer Function (MTF) measurement: ASTM-E 1695-95
- Implementation of (Alternating Direction Method of Multipliers) ADMM optimizor for given energy function using Newton's methd to solve nonconvex subproblem
- Implementation of (Alternating Direction Method of Multipliers) ADMM optimizor for given energy function using Limited-memory BFGS (L-BFGS) method to solve nonconvex subproblem
- Implementation of (Alternating Direction Method of Multipliers) ADMM optimizor for given energy function using ADAM method to solve nonconvex subproblem
- 2D Fourier Transform
- Canny edge detector

Topics

Modulation Transfer Function (MTF) Based on ASTM-E1695-95

Modulation transfer function (MTF) is widely used as a metric for spatial resolution assessment. This project is aimed to implement the MTF measurement of the computed tomography (CT) system based on the norm ASTM-E 1695-95 [1].

CT images of the test object, i.e., a phantom disk made of Aluminium with diameter 20mm, are given. Specifically,

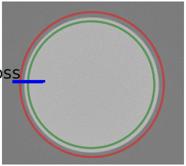
- Three test CT images are given with Input4, Input7, Input10.
- In the file Readme you can find the pixelsize of each CT image, which will be needed when you calculate MTF.
- Compare your MTF curves with the corresponding given MTF curves, noticing the setup parameters: binsize, search distance and fit point count.

Modulation Transfer Function (MTF) Based on ASTM-E1695-95

- Calculate the centre of the phantom in the CT slice.
- Choose inner and outer radii with respect to the centre of circle that bracket the edge.
- Segregate the region between inner and outer radii with bins sized to a small fraction of one pixel.
- Averaging the value of bins according to the distance to the centre.
- Smoothing the averaged curve crossing the edge and do a piece-wise, least-squares cubic fit (ERF).
- Calculate the first derivative of the curve ERF to get PSF.
- Calculate the Fourier Transform of the PSF and normalize the maxima to one (MTF).

Modulation Transfer Function (MTF) Based on ASTM-E1695-95

Profile across
Inner and
outer radii



1. Split 1 pixel to subpixels (according to table in ASTM)
2. Calculate the center and radius of the circle
3. Define the region with outer and inner radii
4. Average the all the profiles in the region
5. Piece-weise least-squares cubic fit
6. Got the white curve beneath

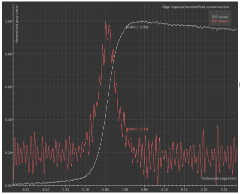
Edge Spread Function (ESF)

First derivative

Point Spread Function (PSF)

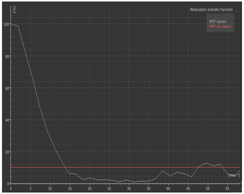
1D Fourier Transform

Modulation Transfer Function (MTF)



Source from tool: VGStudio

1D FFT (padding)



Source from tool: VGStudio

Alternating Direction Method of Multipliers (ADMM)

- An optimization problem solover with good robustness of method of multipliers
- Support decomposition

ADMM (Alternating Direction Method of Multipliers) deals with the following problem [2].

$$\begin{aligned} \textit{minimize} \quad & f(x) + g(y) \\ \textit{subject to} \quad & Ax + By = c \end{aligned} \tag{1}$$

Alternating Direction Method of Multipliers (ADMM)

Here, f, g are assumed convex. An auxiliary variable (Lagrange multiplier) z is introduced to form an function $L_\rho(x, y, z)$

$$L_\rho(x, y, z) = f(x) + g(y) + z^T(Ax + By - c) + \frac{\rho}{2}\|Ax + By - c\|_2^2 \quad (2)$$

where ρ is a tunning parameter. Then, we can iteratively solve for x, y, z in three seperate steps (subproblems):

$$\begin{aligned} x^{k+1} &= \arg\min_x L_\rho(x, y^k, z^k) \\ y^{k+1} &= \arg\min_y L_\rho(x^{k+1}, y, z^k) \\ z^{k+1} &= z^k + \rho(Ax^{k+1} + By^{k+1} - c) \end{aligned} \quad (3)$$

Alternating Direction Method of Multipliers (ADMM)

Proximal operator of a function $f(x)$ is defined as:

$$prox_f(v) = \arg\min_x (f(x) + \frac{1}{2}\|x - v\|_2^2) \tag{4}$$

- Proximal operator is defined in a certain format [3]
- Proximal operator can be solved analytically which accerlates the computation speed

The above mentioned three steps of ADMM can benefit from the proximal operator in terms of computation complexity if a reasonable decomposition of your energy function can be determined so that any step of the three could match the format of proximal operator.

Energy Function

$$J = \min_x \frac{1}{2}\left(\sum_{i=1}^{M} ||y_i - A_i I_0 \exp(-x)||^2_{W_i} + < \log(B_i I_0 \exp(-x) + \sigma_i^2), 1 >\right)$$

$$+ \beta \sum_{p=-w}^{w} \sum_{q=-w}^{w} \gamma(p,q) \parallel x - S_x^p S_y^q x \parallel_1 + \chi_C(x), \quad (5)$$

where $< \cdot >$ indicates a pointwise multiplication of two vectors.
I_0 is a constant. A_i and B_i are constant matrices with size mxn.
S_x, S_y are shift operators along x- and y-axis. σ_i is constant vector.
w is a constant for the window size and β is constant weight.
y_i is the input image with size $mx1$. x is output image with size $nx1$.
M expresses the number of inputs y_i, we make $M = 4$.
W_i is a diagonal weight matrix and can be expressed as

$$W_i = \text{diag}\{\frac{1}{B_{ik} I_0 \exp(-x) + \sigma_{ik}^2}\}.$$

where B_{ik} is the kth row of matrix B_i.

Energy Function

$$J = \min_{x} \frac{1}{2}(\sum_{i=1}^{M} ||y_i - A_i I_0 \exp(-x)||^2_{W_i} + < \log(B_i I_0 \exp(-x) + \sigma_i^2), 1 >)$$

$$+ \beta \sum_{p=-w}^{w} \sum_{q=-w}^{w} \gamma(p,q) \parallel x - S_x^p S_y^q x \parallel_1 + \chi_C(x), \quad (6)$$

Here, $\| \cdot \|_1$ indicates the Euclidean l-1 norm.
We can define $\gamma(p,q) = \alpha^{|p|+|q|}$ where α is a constant.
$\mathcal{X}_B(X)$ is the indicator function of the convex set B which constrains the nonnegativity of the reconstructed X with
$C = \{x : x_K \geq 0, \forall K \in \{1, \ldots, N\}\}$ and

$$\mathcal{X}_{\mathcal{C}}(x) = \begin{cases} 0, & x \subseteq C \\ +\infty, & x \subsetneq C. \end{cases} \quad (7)$$

Energy Function in ADMM

The energy function J can be formulated:

$$J(x,z) = \sum_{i=1}^{M+(2w+1)^2} g_i(z_i) \tag{8}$$

$$\text{subject to } T_i x - z_i = 0, \forall i \in 1 \cdots M + (2w+1)^2$$

with $z_i := \begin{bmatrix} z_{i1} \\ \vdots \\ z_{in} \end{bmatrix}$ and $T := \begin{bmatrix} T_1 \\ \vdots \\ T_{M+(2w+1)^2} \end{bmatrix}$ being a matrix with:

$$T_i = \begin{cases} I_{nxn} & \text{for } i = 1, \dots, M \\ I_{nxn} - S_x^{q(i)} S_y^{p(i)} & \text{for } i = M+1, \dots, M+(2w+1)^2 - 1 \\ I_{nxn} & \text{for } i = M+(2w+1)^2 \end{cases} \tag{9}$$

Energy Function in ADMM

Specially, multiple g_i are defined as follows:

$$g_i(z_i) := \frac{1}{2}(||y_i - A_i I_0 \exp(-z_i)||^2_{W_i} + < \log(B_i I_0 \exp(-z_i) + \sigma_i^2), 1 >)$$
$$\text{for } i = 1, \ldots, M$$
$$g_i(z_i) := \beta\gamma(p, q)||z_i||_1 \text{ for } i = M+1, \ldots, M + (2w+1)^2 - 1$$
$$g_i(z_i) := \chi_C(z_i) \text{ for } i = M + (2w+1)^2 \tag{10}$$

Note that

$$< \log(B_i I_0 \exp(-z_i) + \sigma_i^2), 1 >= \sum_{k=1}^{n} \log(I_0 < [B_i]_k, \exp(-z_i) > + [\sigma_i]_k^2).$$

Energy Function in ADMM

The Lagrangian function of $J(x, z)$ and the constraint $T_i x - z_i = 0$ is

$$\mathcal{L}(x, z, p) = \sum_{i=1}^{M+(2w+1)^2} (g_i(z_i) + < p_i, T_i x - z_i >). \tag{11}$$

The augmented Lagrangian function is

$$\mathcal{L}_{H_k}(x, z, p) := \sum_{i=1}^{M+(2w+1)^2} (g_i(z_i) + < p_i, T_i x - z_i > + \frac{1}{2} || T_i x - z_i ||_{H_{ik}}^2)$$

where matrix H_{ik} is defined:

$$H_{ik} := \text{diag}[\rho_i, \ldots, \rho_i], \forall i \in 1, \cdots, M + (2w+1)^2 \tag{12}$$

with ρ_i being repeated according to the dimension of the variable z_i .

Energy Function in ADMM

In our case we have $M = 4$ low resolution images and the following iteration scheme:

$$
\begin{aligned}
x^{k+1} &= \operatorname*{arg\,min}_{x \in \mathcal{X}} \sum_{i=1}^{M+(2w+1)^2} \frac{\rho_i}{2} || T_i x - z_i^k + \frac{p_i^k}{\rho_i} ||_2^2 \\
z_i^{k+1} &= \operatorname*{arg\,min}_{z_i \in \mathcal{Z}_i} g_i(z_i) + \frac{\rho_i}{2} || z_i - T_i x^{k+1} - \frac{p_i^k}{\rho_i} ||_2^2 \\
p_i^{k+1} &= p_i^k + \rho_i (T_i x^{k+1} - z_i^{k+1}).
\end{aligned} \tag{13}
$$

The update of x^{k+1} can be solved by e.g. conjugate gradient.
To update $z_i^{k+1}, i \in [1, \cdots, M]$, $g_i(z_i)$ will be solved using e.g. Newton's method, L-BFGS and ADAM and for the rest $z_i^{k+1}, i \in \left[M+1, \cdots, M+(2w+1)^2\right]$, $g_i(z_i)$ can be calculated using Proximal operator (See following slides).

Energy Function in ADMM

To update x^{k+1} using the conjugate gradient, the partial gradient of $V_i(x) := \frac{\rho_i}{2}||T_i x - z_i^k + \frac{p_i^k}{\rho_i}||_2^2$ must be calculated by

$$\nabla V_i(x) = \rho_i T_i^T (T_i x - z_i^k + \frac{p_i^k}{\rho_i}), \tag{14}$$

where ρ_i is a tunable user-define scalar.
To update z_i^{k+1}, the partial gradient and Hessian (for Newton's method) have to be calculated as below (See following slides).

Hadamard Product

We define the product between a matrix and a vector notated by $\odot$:

$$A_{mxn} \odot \vec{b}_{mx1} = \begin{pmatrix} a_{11} \cdot b_1, & \cdots, & a_{1n} \cdot b_1 \\ \vdots & \ddots & \vdots \\ a_{m1} \cdot b_m, & \cdots, & a_{mn} \cdot b_m \end{pmatrix}$$

$$\vec{b}_{mx1} \odot \vec{b}_{mx1} = \begin{pmatrix} b_1 \cdot b_1 \\ \vdots \\ b_m \cdot b_m. \end{pmatrix}$$

It can be easily implemented by:

$$A_{mxn} \odot \vec{b}_{mx1} = \text{diag}\left[b_1, \cdots, b_m\right] \cdot A_{mxn}$$

$$b_{mx1} \odot \vec{b}_{mx1} = \text{diag}\left[b_1, \cdots, b_m\right] \cdot \vec{b}_{mx1}$$

Partial Derivative of Weighted L2

For some nonconvex $g_i(z_i)$, i.e., $g_{1,\cdots,M}$, one needs to use, e.g., Newton's method, quasi-Newton's method or adaptive moment estimation (ADAM) to solve it.
We compute the partial derivative of $G_i(z_i)$ and $H_i(z_i)$:

$$G_i(z_i) := \frac{1}{2}||y_i - A_i I_0 \exp(-z_i)||^2_{W(z_i)} = \frac{1}{2}\sum_{k=1}^{n} \frac{(y_{ik} - A_{ik} I_0 \exp(-z_i))^2}{B_{ik} I_0 exp(-z_i) + \sigma_{ik}^2}$$

$$\frac{\partial G_i}{\partial z_{ij}}(z_i) = \sum_{k=1}^{n} \Bigg[\frac{(y_{ik} - A_{ik} I_0 \exp(-z_i)) A_{ikj} I_0 \exp(-z_{ij})}{(B_{ik} I_0 \exp(-z_i) + \sigma_{ik}^2)} + \frac{1}{2} \frac{B_{ikj} I_0 exp(-z_{ij}) \left(y_{ik} - A_{ik} I_0 \exp(-z_{ij})\right)^2}{(B_{ik} I_0 exp(-z_i) + \sigma_{ik}^2)^2} \Bigg] \tag{15}$$

Here j means the jth element in the vectorized z_i and A_{ik} indicates the $k$the row of the matrx A_i.

Partial Derivative

If we formulate in matrixwise, we have:

$$\frac{\partial G_i}{\partial z_i}(z_i) = I_0 A_i^T M_i \odot exp(-z_i) + \frac{1}{2} I_0 B_i^T M_i^2 \odot exp(-z_i) \tag{16}$$

where $M_i = \dfrac{y_i - A_i I_0 \exp(-z_i)}{B_i I_0 exp(-z_i) + \sigma_i^2}$ is the elementwise division and $M_i^2 = M_i \odot M_i$ is the elementwise square of M_i.
In addition, to update z_i^{k+1}, the partial derivative of $U_i(z) := \dfrac{\rho_i}{2} ||z_i - T_i x^{k+1} - \dfrac{p_i^k}{\rho_i}||_2^2$ must be calculated:

$$\nabla U_i(z_i) = \rho_i(z_i - T_i x^{k+1} - \frac{p_i^k}{\rho_i}) \tag{17}$$

Partial Derivative of Log Term

If we notate: $H_i(z_i) := \frac{1}{2} < \log(B_i I_0 \exp(-z_i) + \sigma_i^2), 1 >$

$$\frac{\partial h_i}{\partial z_{ij}}(z_i) = \frac{1}{2}\sum_{k=1}^{n}\left[-\frac{1}{B_{ik} I_0 \exp(-z_i) + \sigma_{ik}^2} \cdot B_{ikj} I_0 exp(-z_{ij})\right] \tag{18}$$

In matrixwise, we have:

$$\frac{\partial H_i}{\partial z_i}(z_i) = -\frac{1}{2} I_0 B_i^T M_i \odot exp(-z_i) \tag{19}$$

where $M_i = \frac{1}{B_i I_0 \exp(-z_i) + \sigma_i^2}$ is the elementwise division.
Note that the operations between vectors are elementwise and operations between vector and matrix should be commonly computed.

Hessian Matrix of Weighted L2-Norm

The Hessian matrix of $G_i(z_i)$can be calculated by:

$$\mathcal{H}(G_i(z_i)) = \text{diag}\left[-\exp(-z_{i1}), \dots, -\exp(-z_{in})\right] \odot$$
$$\left(I_0 A_i^T \cdot \frac{(y_i - A_i I_0 \exp(-z_i))}{B_i I_0 \exp(-z_i) + \sigma_i^2} + \frac{1}{2} I_0 B_i^T \cdot \frac{(y_i - A_i I_0 \exp(-z_i))^2}{(B_i I_0 \exp(-z_i) + \sigma_i^2)^2} \right) +$$
$$\text{diag}[\exp(-z_{i1}), \dots, \exp(-z_{in})] \cdot$$
$$\left(I_0^2 A_i^T (A_i \odot \frac{1}{I_0 B_i \exp(-z_i) + \sigma_i^2}) + (I_0^2 A_i^T (B_i \odot \frac{y_i - A_i I_0 \exp(-z_i)}{(I_0 B_i \exp(-z_i) + \sigma_i^2)^2}) + \right.$$
$$\left. I_0^2 B_i^T (A_i \odot \frac{y_i - A_i I_0 \exp(-z_i)}{(I_0 B_i \exp(-z_i) + \sigma_i^2)^2}) + I_0^2 B_i^T (B_i \odot \frac{(y_i - A_i I_0 \exp(-z_i))^2}{(I_0 B_i \exp(-z_i) + \sigma_i^2)^3}) \right)$$
$$\cdot \text{diag}[\exp(-z_{i1}), \dots, \exp(-z_{in})]$$

Hessian Matrix of Log Term

The Hessian matrix of $H_i(z_i)$ can be formulated by:

$$\mathcal{H}(H_i(z_i)) = \frac{1}{2}\text{diag}\left[\exp(-z_{i1}), \dots, \exp(-z_{in})\right] \odot$$
$$\left(I_0 B_i^T \cdot \frac{1}{I_0 B_i \exp(-z_i) + \sigma_i^2}\right) + \text{diag}\left[\exp(-z_{i1}), \dots, \exp(-z_{in})\right] \cdot$$
$$I_0^2 (B_i^T (B_i \odot \frac{1}{(I_0 B_i \exp(-z_i) + \sigma_i^2)^2})) \odot \exp(-z_i) \cdot$$
$$\text{diag}[\exp(-z_{i1}), \dots, \exp(-z_{in})]$$

The last needed Hessian is $U_i(z_i) = \frac{\rho_i}{2}||z_i - T_i x^{k+1} - \frac{p_i^k}{\rho_i}||_2^2$, which is computed rather easily

$$\mathcal{H}(U_i(z_i)) = \text{diag}[\rho_i, \dots, \rho_i] \tag{20}$$

Proximal Operators

For $g_i(z_i)$ with $i = M+1, \dots, M+(2w+1)^2$, closed form solution exist and can be computed analytically with the proximal operator:

$$\arg\min_{z_i \in \mathcal{Z}_i} \beta\gamma(p,q)||z_i||_1 + \frac{\rho_i}{2}||z_i - x^k - \frac{p_i^k}{\rho_i}||_2^2$$

$$= prox_{\beta\gamma(p,q)(\rho_i)^{-1}||\cdot||_1}(x^k + \frac{p_i^k}{\rho_i})$$

$$= \begin{cases} [x^k + \frac{p_i^k}{\rho_i}]_l - \beta\gamma(p,q)(\rho_i)^{-1}, & \text{if } [x^k + \frac{p_i^k}{\rho_i}]_l \geq \beta\gamma(p,q)(\rho_i)^{-1} \\ 0, & \text{if } |[x^k + \frac{p_i^k}{\rho_i}]_l| \leq \beta\gamma(p,q)(\rho_i)^{-1} \\ [x^k + \frac{p_i^k}{\rho_i}]_l + \beta\gamma(p,q)(\rho_i)^{-1}, & \text{if } [x^k + \frac{p_i^k}{\rho_i}]_l \leq -\beta\gamma(p,q)(\rho_i)^{-1} \end{cases}$$

Here l means elementwise comparison.

Proximal and Shift Operators

The proximal operator for the last $g_i(z_i)$ can be computed by

$$z_i^{k+1} = \max(x^{k+1} + \frac{p_i^k}{\rho_i}, 0) \text{ for } i = M + (2w+1)^2.$$

The shift operators S_x and S_y have been given in the code in function *Mmatrix*(). The scalar parameter p and q should be passed to the arguments *deltaX* and *deltaY* which specify the offset in X- and Y-direction.

Initialization Parameters

For the current status, the initialization parameters can be set as:
z_i^0, for $i \in \left[1, \cdots, M + (2w+1)^2\right]$, we say $I_0 exp(-z_i) = F_{bic}(y_i)$ which is the bicubic interpolation of the input image y_1.
p_i^0, for $i \in \left[1, \cdots, M + (2w+1)^2\right]$ is zero matrix with the same size as x.
x^0 is the zero matrix (or bicubic interpolation of y_2).
ρ_i, for $i \in \left[1, \cdots, M + (2w+1)^2\right]$ is some constant value large.
You can tune all the initialization parameters by yourself.

ADMM with Newton's Method

Although ADMM is originally designated for convex problem. It is robust enough even for some nonconvex problems.

For the nonconvex subproblems of ADMM, one could use Newton's method to solve them [4]. For the other subproblems, one could take advantage of proximal operator.

For Newton's method, you need to compute the gradient and inverse Hessian of the energy functon on X. The Hessian matrix is given in formula. The inverse of the Hessian should not be calculated directly but approximated by , e.g., factorization or iterative method like conjugate gradient.

In the code given, only the matrices of A_i, A_i^T, B_i and B_i^T are calculated and saved in viennacl::sparseMatrix.

ADMM with L-BFGS

For the nonconvex subproblems of ADMM, one could use a quasi-Newton's method L-BFGS to solve them [5][6].

For the other subproblems, one could take advantage of proximal operator.

For L-BFGS, an estimation for the inverse Hessian is calculated to reduce the computation load. Instead of calculating the inverse of the Hessian matrix, the previous X and gradients are saved to estimate the inverse of the Hessian which reduces the compulation complexity dramatically.

In the code given, only the matrices of A_i, A_i^T, B_i and B_i^T are calculated and saved in viennacl::sparseMatrix.

ADMM with ADAM

For the nonconvex subproblems, one could use adaptive moment estimation (ADAM)to solve them [7].

- A method for stochastic optimation which combines the advantages of two stochastic gradient descent methods AdaGrad and RMSProp.
- An algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. Specifically, it updates the stepsize not only based on the average first moment (the mean) as in RMSProp, but also making use of the average of the second moments of the gradients (the uncentered variance).

In the code given, only the matrices of A_i, A_i^T, B_i and B_i^T are calculated and saved in viennacl::sparseMatrix.

2D Fast Fourier Transform

Implement a Fast Fourier Transform on the GPU. Support for non-power-of-two input sizes is optional.

Canny Edge Detector

Implement the Canny edge detector on the GPU. The program should include graphical output (e.g. using OpenGL). There also should be an option to output the various intermediate stages.

Sources

1 Standard Test Method for Measurement of Computed Tomography (CT) System Performance.

2 Alternating Direction Method of Multipliers.

3 Proximal Algorithms.

4 Newton's Method for Unconstrained Optimization.

5 Quasi-Newton methods.

6 Optimization methods.

7 ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION.

8 Statistical Image Reconstruction Using Mixed Poisson-Gaussian Noise Model for X-Ray CT.