Code documentation
Version 1.4

Julian Rasch

June 15, 2017

1 The anatomical prior and its reconstruction

We shall start with an explanation of the acquisition and reconstruction scheme for the anatomical prior.

1.1 Operator modeling and the method

In order to clarify the code, we need to further specify the sampling operator for the anatomical prior. We follow the ideas of [Ehrhardt, Betcke]. First, and most important, we restrict the prior to be real-valued (otherwise it is not directly clear how to define gradients, subgradients etc. for the reconstruction, see remark below), i.e. $u_0 \in \mathbb{R}^{N_0}$. However, since the Fourier transform acts on complex-valued images, we need to embed our image into the complex numbers $\mathbb{C}^{N_0}$ via the embedding operator

$$i\colon \mathbb{R}^{N_0} \to \mathbb{C}^{N_0}, u_0, \quad u \mapsto u + 0i.$$

It is the adjoint operator of the real part restriction, i.e. $i^*(x+iy) = \mathrm{Re}(x+iy) = x$, for $x, y \in \mathbb{R}^{N_0}$. We then apply a standard Cartesian Fourier transform to the embedded image. In order to simulate an arbitrary MR acquisition protocol, we do not use all the Fourier coefficients, but a sampling operator $\mathcal{S}_0\colon \mathbb{C}^{N_0} \to \mathbb{C}^{M_0}$ to only choose a subset of the Fourier coefficients. For practical use, only the situation of undersampling, i.e. $M_0 \ll N_0$ is relevant. Hence, if we let $\mathcal{P}\colon \{1,\dots,M_0\} \to \{1,\dots,N_0\}$ be a mapping which chooses M_0 Fourier coefficients from the N_0 coefficients available, we can define the sampling operator $\mathcal{S}_0$ applied to $z \in \mathbb{C}^{N_0}$

$$\mathcal{S}_0\colon \mathbb{C}^{N_0} \to \mathbb{C}^{M_0}, \quad (\mathcal{S}_0 x)_k = z_{\mathcal{P}(k)}.$$

The full forward operator $\mathcal{K}_0$ can hence be expressed as

$$\mathcal{K}_0\colon \mathbb{R}^{N_0} \xrightarrow{i} \mathbb{C}^{N_0} \xrightarrow{\mathcal{F}} \mathbb{C}^{N_0} \xrightarrow{\mathcal{S}_0} \mathbb{C}^{M_0}. \tag{1}$$

Its adjoint is

$$\mathcal{K}_0^*\colon \mathbb{C}^{M_0} \xrightarrow{\mathcal{S}_0^*} \mathbb{C}^{N_0} \xrightarrow{\mathcal{F}^{-1}} \mathbb{C}^{N_0} \xrightarrow{i^*} \mathbb{R}^{N_0}, \tag{2}$$

where $\mathcal{S}_0^*$ 'fills' the missing frequencies with zeros, i.e.

$$\mathcal{S}_0^*(z) = \sum_{k=1}^{M_0} z_k \delta_{k,\mathcal{P}(k)}.$$

Remark 1. *Note that applying $\mathcal{S}_0^*$ is equivalent to the minimum norm solution restricted to the real part, hence can be used as a 'standard inversion' for comparison.*

In order to get a piecewise constant reconstruction of the prior to use its gradient information, we perform a total variation regularized reconstruction.

$$\min_{u_0 \in \mathbb{R}^{N_0}} \frac{\alpha_0}{2} \|\mathcal{K}_0 u_0 - f_0\|_{\mathbb{C}}^2 + \mathrm{TV}(u_0), \tag{3}$$

where $\mathrm{TV}(u_0) = \|\nabla u_0\|_1$ with a discrete gradient operator $\nabla\colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_0 \times 2}$ (cf. [Chambolle]). Hence

$$\mathrm{TV}(u_0) = \|\nabla u_0\|_1 = \sum_{k=1}^{N_0} \sqrt{|\nabla u_0|_{k,1}^2 + |\nabla u_0|_{k,2}^2}.$$

After the reconstruction, we can update the subgradient via the optimality condition:

$$p_0 = \frac{1}{\alpha_0} \mathcal{K}_0^*(f_0 - \mathcal{K}_0 u_0). \tag{4}$$

Remark 2. *It is also possible to define the operator $\mathcal{K}_0$ not on $\mathbb{R}^{N_0}$ but on $\mathbb{C}^{N_0}$, and perform the reconstruction with the constraint that u_0 is real-valued. However, then there is no accessible optimality condition which allows to get a subgradient.*

A general problem with this subgradient is, due to its relation to the residual, it contains noise. For standard Bregman iterations this is desirable [Burger, Osher], however, here it is questionable, why 'noise' should be transferred from the prior to the following dynamic reconstruction. Therefore, we provide an alternative solution, an artificial 'subgradient'. Recall, that $p_0 \in \partial \mathrm{TV}(u_0)$ if and only if there exists $q_0 \in \partial \|\nabla u_0\|_1$ such that $p_0 = -\mathrm{div}(q_0)$. In particular we have that

$$q_0 \begin{cases} = \nabla u_0 / |\nabla u_0|, & \text{if } \nabla u_0 \neq 0, \\ \in B(0,1), & \text{else.} \end{cases}$$

Hence, having a good reconstruction u_0 of the prior at hand, we can create an artificial one by letting

$$\tilde{q}_0 = \begin{cases} \nabla u_0 / |\nabla u_0|, & \text{if } \nabla u_0 \neq 0, \\ 0, & \text{else.} \end{cases}$$

However, in view of not entirely perfect reconstructions in the numerical situation, it is preferable to decide, which height a jump has to have to become an edge. Hence, introducing the edge parameter η, we have

$$\tilde{q}_{0,\eta} = \begin{cases} \nabla u_0 / |\nabla u_0|, & \text{if } |\nabla u_0| \geq \eta, \\ 0, & \text{else.} \end{cases}$$

Then, finally, $p_0 = -\mathrm{div}(q_0)$ or $\tilde{p}_{0,\eta} = -\mathrm{div}(\tilde{q}_{0,\eta})$, respectively. We shall however simply refer to the subgradient as p_0, independently of the choice of q_0. Comparing the different choices of gradients we notice, that the 'artificial' subgradients are basically a 'clean' version of p_0.

1.2 Numerical realization

The above considerations can be found in this code toolbox. We shall start with the operator $\mathcal{K}_0$ and its adjoint $\mathcal{K}_0^*$.

- `sampling_geom.m`: This function can create a lot of possible sampling pattern for the MR as a binary map, i.e. represents the operator $\mathcal{S}_0$. For the prior, good choices are a `'full'` sampling, or a sufficiently dense sampling on radial `'spokes'` through the k-space center. The sampling can be created with the call `S_0 = sampling_geom(u0,'full');` or `S_0 = sampling_geom(u0,'spokes1','nSpokes',n);`, where `n` is the desired number of spokes. Other samplings can be called identically, just check the documentation of the function, including a lot of options like number of sampling points and many more. The input `u0` is only used for the size of the Fourier transform, hence can also be a dummy.

- `sampling_op.m`: Once you have generated a sampling for $\mathcal{S}_0$ via `sampling_geom.m`, you can create a function handle for the operator $\mathcal{K}_0$, i.e. a Fourier transform followed by a projection onto the frequencies specified in $\mathcal{S}_0$ (or `S_0`). The syntax is `K_0 = @(x)sampling_op(x,S_0);`.
- `sampling_adj.m`: This function is the adjoint operator $\mathcal{K}_i^*$, and can be called analogously.
- `sampling_vis.m`: This function can visualize the sampling via `sampling_vis(S_0);`.

If we use artificial data made from a phantom, we can use the forward operator $\mathcal{K}_0$ to create some data f_0 from the clean phantom u_0^{clean}.

- `fourier_data.m`: This function creates noisy and undersampled Fourier data from a clean phantom u_0^{clean}. The ratio of energy of the noise and energy of the data can be chosen by the parameter `energy` . The function call is `f0 = fourier_data(K_0{1,1},u0_clean,energy);`. You can also visualize the data and how it is sampled via `sampling_vis(f0);`.

Finally we can do the reconstruction (3).

- `l2_tv.m`: This function solves (3) with a primal-dual scheme and monitors the decrease of the primal energy as well as the primal-dual residual. We can set a few options via the script `paramfile_tv.m`.

```
function param = paramfile_tv

param           = struct;
param.norm_type = 'iso';
param.pdtol     = 1e-5;
param.niter     = 5e4;
param.int       = 100;

param.tau       = 0.3;
param.sigma     = 0.3;

param.show      = true;

end
```

  The stopping criterion is a sufficiently small primal-dual residual [Ref], we can increase the accuracy of the solution by lowering the tolerance `pdtol`. The value of `niter` controls the amount of iterations, while `int` controls the intervals in which the primal energy and the primal-dual residual is evaluated. If `show` is set to `true`, the function plots the progress of the energy and the residual after convergence. The parameters `tau` and `sigma` are the step sizes for the algorithm.

Remark 3. *We would like to comment on the choice of the step sizes τ and σ. In theory, they need to be chosen such that the condition $\tau\sigma L^2 < 1$ is fulfilled, where L denotes the operator norm of the linear operator in the formulation of the used primal-dual scheme (cf. [Chambolle-Pock]). However, in our case, especially when using function handles and operators, that are not available in matrix form, it is not entirely clear how to determine L. This is in particular difficult for the algorithm in the next section, which involves a combination of block operators (cf. [Rasch] for a version with two blocks). Hence, we determine `tau` and `sigma` empirically. We just choose a value and control the convergence, if it oscillates, we decrease both parameters by a factor and try again. However, I am working on a method to determine it automatically.*

After convergence, we need to compute a subgradient, either via the optimality condition, or an artificial one. This is done via the function `artificial_subgradient.m`. The whole procedure is summarized in the following code:

```
%% Create sampling operators and an artificial data set for the prior

% Create a sampling S_0 for the anatomical prior (here: full k-space)
S_0  = sampling_geom(u0_clean, 'full');

% Setup the operator K_0 and its adjoint
K_0{1,1} = @(x) sampling_op(x,S_0);
K_0{1,2} = @(y) sampling_adj(y,S_0);

% Create data f_0 with complex valued Gaussian noise
% of the anatomical prior
energy = 0.05; % noise level
f0 = fourier_data(K_0{1,1},u0_clean,energy);

% Visualize the sampling
% sampling_vis(f0);

%% Do a TV reconstruction of the prior u0

param_u0  = paramfile_tv; % new parameter file
alpha_u0  = 75; % regularization parameter
[u0, hist_u0] = l2_tv(f0,alpha_u0,K_0,'param',param_u0,'show',false);

% Compute the true subgradient from the optimality condition
p_true = K_0{1,2}(alpha_u0 * (f0 - K_0{1,1}(u0)));

% Create an artificial subgradient with edge parameter 'eta'
eta = 0.025;
p_art = artificial_subgradient(u0,'eta',eta);
```

It is interesting to visually compare `p_true` and `p_art`, in particular it gives an indicator of how big we need to choose the edge parameter `eta`. Just make sure that `p_art` resembles `p_true` without noise.

2 The dynamic scan

Now we can proceed to the dynamic scans, which basically follow the same idea as before. The operator $\mathcal{K}_t$ and its adjoint $\mathcal{K}_t^*$ can be defined analogously via

$$\mathcal{K}_t \colon \mathbb{R}^N \xrightarrow{i} \mathbb{C}^N \xrightarrow{\mathcal{F}} \mathbb{C}^N \xrightarrow{\mathcal{S}_t} \mathbb{C}^{M_t} \tag{5}$$

and

$$\mathcal{K}_t^* \colon \mathbb{C}^{M_t} \xrightarrow{\mathcal{S}_t^*} \mathbb{C}^N \xrightarrow{\mathcal{F}^{-1}} \mathbb{C}^N \xrightarrow{i^*} \mathbb{R}^N. \tag{6}$$

Remark 4. *We would like to remark that we keep N fixed for the dynamic scans, implying that the resolution of the u_t stays the same over time. We however theoretically keep the possibility that $N_0 > N$, i.e. that the prior has a higher resolution. This would involve some downsampling techniques after the computation of the subgradient p_0. Furthermore, in general M_t will be much smaller than M_0 such that the acquisition is sufficiently fast.*

The reconstruction of the dynamic series then can be done via the following reconstruction scheme:

$$\arg\min_{\mathbf{u}} \sum_{t=1}^{T} \frac{\alpha_t}{2}\|\mathcal{K}_t u_t - f_t\|_{\mathbb{C}}^2 + w_t \mathrm{TV}(u_t) + (1-w_t)\mathrm{ICB}_{\mathrm{TV}}^{p_0}(u_t, u_0) + \sum_{t=1}^{T-1} \frac{\gamma_t}{2}\|u_{t+1} - u_t\|^2, \tag{7}$$

where $\mathbf{u} = [u_1, \dots, u_T]$ and $u_t \in \mathbb{R}^N$ for all $t = 1, \dots, T$. We need to choose three sets of parameters: the regularization parameters α_t and γ_t, which control the tradeoff between data fidelity, temporal regularization, and $w_i \in [0, 1]$, which controls the amount of total variation regularization versus similarity to the prior. They can be chosen equally for all time steps t, but can as well be adapted to when the activation is expected.

2.1 Creating the operators

The operators can be created the same way as above. The difference is that we now want to perform undersampling, i.e. we need to pick a different sampling scheme. A particularly interesting one is a sampling on radial spokes that are chosen according to the golden ratio [ref]. More precisely, we divide π by the golden ratio $r = \frac{1+\sqrt{5}}{2}$ to obtain

$$\frac{\pi}{r} = \frac{2 * \pi}{1 + \sqrt{5}} \approx 1.9416.$$

Then we choose the k-th spoke according to the angle

$$\varphi_k = \frac{k\pi}{r} \mod 2\pi.$$

This consecutively fills the entire k-space in an interleaving fashion. For example, choosing 10 spokes per time step, we use spokes with the angles $\{\varphi_1, \dots, \varphi_{10}\}$ for time step $t = 1$, and continue with angles $\{\varphi_{11}, \dots, \varphi_{20}\}$ for $t = 2$ and so on. The same way as before, we can also create noisy data according to the respective sampling. We summarize the procedure in the following code:

```
%% Create samplings S and data f for the dynamic phantom u
for t = 1:T
    % Samplings
    nSpokes     = 18;
    S{t}        = sampling_geom(u_clean{t}, 'spokes5','nSpokes',...
        nSpokes,'number',(t-1)*nSpokes + 1);
    K{t,1}      = @(x) sampling_op(x,S{t});
    K{t,2}      = @(y) sampling_adj(y,S{t});
    % Data
    f{t} = fourier_data(K{t,1},u_clean{t},energy);
end

% Show two consecutive samplings, and their overlap
figure;
subplot(131); sampling_vis(S{1},1);
subplot(132); sampling_vis(S{2},1);
subplot(133); sampling_vis(S{1} | S{2},1);
```

If T is high, i.e. the number of time steps, for computational reasons it is smarter to divide the data set into *ascending and overlapping* bits T_l, i.e. $T_1 = \{1, \dots, 10\}$, $T_2 = \{10, \dots, 20\}$ and so on, such that we solve

$$\arg\min_{\mathbf{u}} \sum_{t\in T_l} \frac{\alpha_t}{2}\|\mathcal{K}_t u_t - f_t\|_{\mathbb{C}}^2 + w_t \mathrm{TV}(u_t) + (1-w_t)\mathrm{ICB}_{\mathrm{TV}}^{p_0}(u_t, u_0) + \sum_{t\in T_l \setminus \max\{T_l\}} \frac{\gamma_t}{2}\|u_{t+1} - u_t\|^2. \tag{8}$$

We can then choose the reconstruction parameters for every set (note that this is an arbitrary choice here, and that the regularization parameters can of course also vary within the sets).

```
%% Divide the data sets into overlapping bits
set{1} = 1:10;
set{2} = 10:20;
set{3} = 20:30;
set{4} = 30:40;
set{5} = 40:50;

% Choose the reconstruction parameters
w{1} = 0.2 * ones(numel(set{1}),1);
w{2} = 0.3 * ones(numel(set{2}),1);
w{3} = 0.4 * ones(numel(set{3}),1);
w{4} = 0.3 * ones(numel(set{4}),1);
w{5} = 0.2 * ones(numel(set{5}),1);

alpha{1} = 500 * ones(numel(set{1}),1);
alpha{2} = 500 * ones(numel(set{2}),1);
alpha{3} = 500 * ones(numel(set{3}),1);
alpha{4} = 500 * ones(numel(set{4}),1);
alpha{5} = 500 * ones(numel(set{5}),1);

gamma{1} = 75 * ones(numel(set{1}),1);
gamma{2} = 75 * ones(numel(set{2}),1);
gamma{3} = 75 * ones(numel(set{3}),1);
gamma{4} = 75 * ones(numel(set{4}),1);
gamma{5} = 75 * ones(numel(set{5}),1);
```

The last thing to do is to run the reconstruction:

```
%% Run the reconstruction
for i = 1:numel(set)
    [u{i},hist_u{i}] = jr_fmri(f(set{i}), alpha{i}, gamma{i}, w{i}, K(
       set{i},:), p_art,'param.pdtol',5e-02, 'show',false);
end

% Check the convergence (e.g. for the first set{i})
figure;
subplot(121); plot(hist_u{1}.en); title('Primal energy');
subplot(122); plot(hist_u{1}.pdres); title('PD residual');
```

Here, we lowered the tolerance for 'convergence' of the primal-dual residual. For a higher accuracy we can change this for the cost of a longer computation time. If you have time, check that both energy and residual are very small. If energy and residual oscillate strongly, decrease the step sizes a little.

3 Evaluation

This section is dedicated to the evaluation of the results. I have some code available, which is however not perfect yet, and maybe not needed exactly now. I will add this as soon as we have figured out a nice way to visualize everything.

4 Implementation details

This section will to contain the details of the implementation, such as the primal-dual formulation and some further description of how the algorithms `l2_tv.m` and `jr_fmri.m` are realized and implemented. This is still to do.