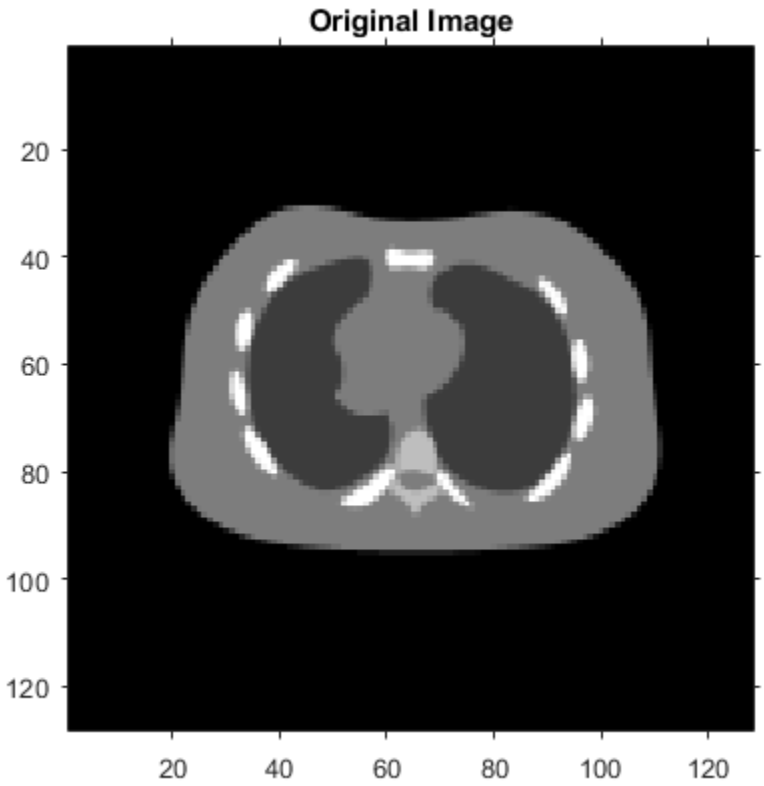
Table of Contents

Prepping the script

```
clc; clear; close all;
addpath('../functions/');
iptsetpref('ImshowAxesVisible','on');
iptsetpref('ImshowInitialMagnification','fit');
addpath('../functions/');
img = double(imread('../../data/ChestPhantom.png'))/255;
figure;
imshow(img);
title('Original Image');
n = size(img, 1);
theta = 0:179;
```



Building system matrix A

Explanation: Ax = b We choose an image x such that exactly one pixel has value 1 and all other pixels have value 0. We then compute radon transform of this image x using Matlab's 'radon()' function. Now this computed transform corresponds to a column of the matrix A. Following this procedure we generate all columns of A and hence the entire matrix A.

```
[rad, xp] = radon(img, theta);
A = zeros(size(rad,1)*size(rad,2),n*n);
for i = 1:n*n
    x = zeros(n,n);
    x(i)=1;
    [rad_temp, xp_temp] = radon(x, theta);
    A(:,i) = rad_temp(:);
end
A = sparse(A);
```

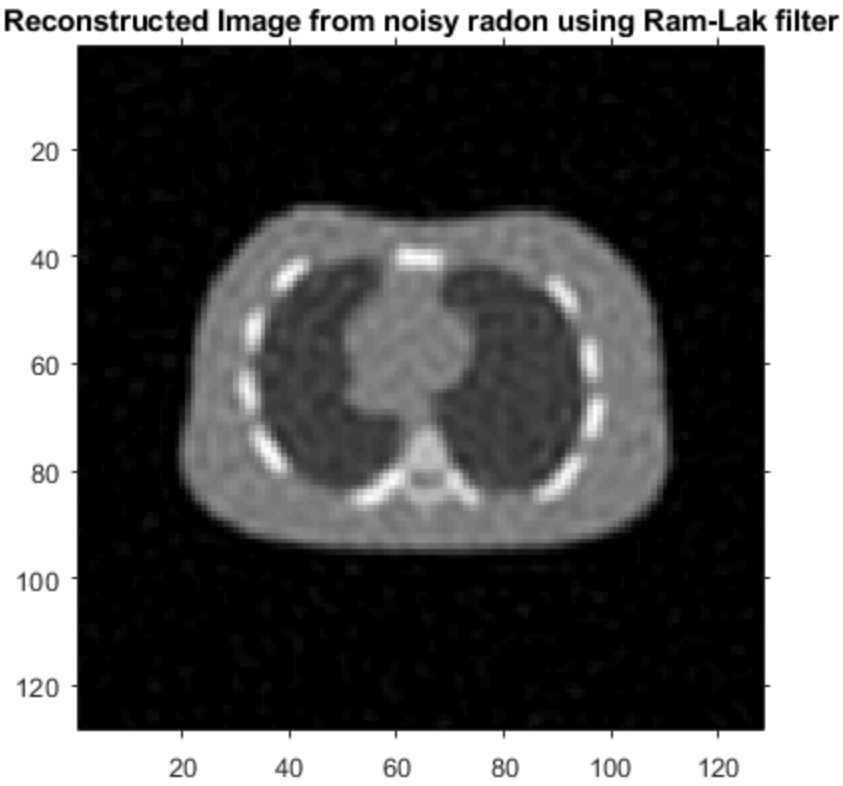
Forward projection using A

```
radon_trans = A*img(:);
radon_trans = reshape(radon_trans,size(rad));
rang = max(radon_trans(:))-min(radon_trans(:));
noised_radon = radon_trans + randn(size(radon_trans))*0.02*rang;
```

Filtered backprojection reconstruction

```
img_filter = iradon(noised_radon,theta,'linear','Ram-Lak',0.5,n);
figure;
imshow(img_filter);
title('Reconstructed Image from noisy radon using Ram-Lak filter');
error = sqrt(sum((img_filter(:)-img(:)).^2)/sum(img(:).^2));
fprintf("RRMSE Ram-Lak filter %.4f\n",error);
```

```
RRMSE Ram-Lak filter 0.1338
```

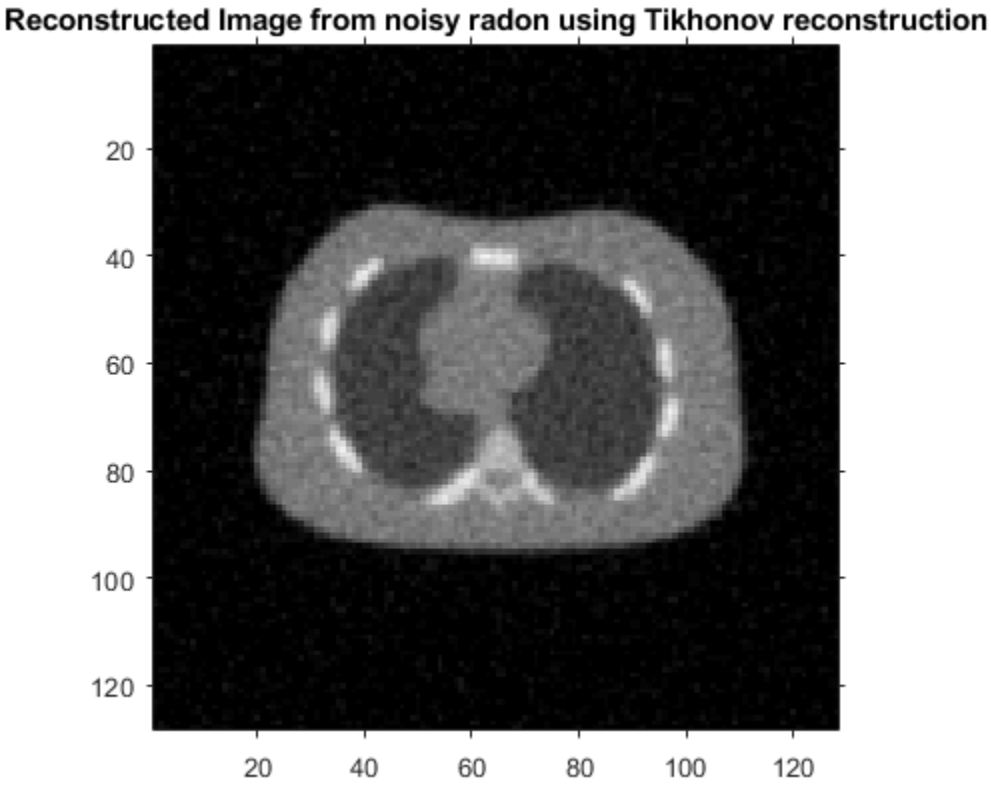


Tikhonov regularized reconstruction

```
fprintf('Tikhonov Regularized Reconstruction\n');
alpha_opt = 144; gamma_opt = 1; eps = 1e-8; epochs = 200;
alpha = alpha_opt; gamma = gamma_opt;

[tikh_img, ~] = descent(A, noised_radon(:), eps, alpha, gamma, epochs,
 @tikh_grad, @tikh_cost);

% RRMSE
error = sqrt(sum((tikh_img-img(:)).^2)/sum(img(:).^2));

% Optimal Parameters Testing
fprintf('alpha(a) = %.3f\n', alpha_opt);
fprintf('RRMSE(a) = %f\n', error);

alpha = 0.8 * alpha_opt; gamma = gamma_opt;
[tikh_img1, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @tikh_grad, @tikh_cost);
error = sqrt(sum((tikh_img1-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(0.8a) = %f\n', error);

alpha = 1.2 * alpha_opt; gamma = gamma_opt;
[tikh_img1, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @tikh_grad, @tikh_cost);
```

```
error = sqrt(sum((tikh_img1-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(1.2a) = %f\n', error);

% Plot image reconstructed
img_recon = reshape(tikh_img,[n,n]);
figure;
imshow(img_recon);
title('Reconstructed Image from noisy radon using Tikhonov
 reconstruction');
```

```
Tikhonov Regularized Reconstruction
alpha(a) = 144.000
RRMSE(a) = 0.161286
RRMSE(0.8a) = 0.163049
RRMSE(1.2a) = 0.162028
```

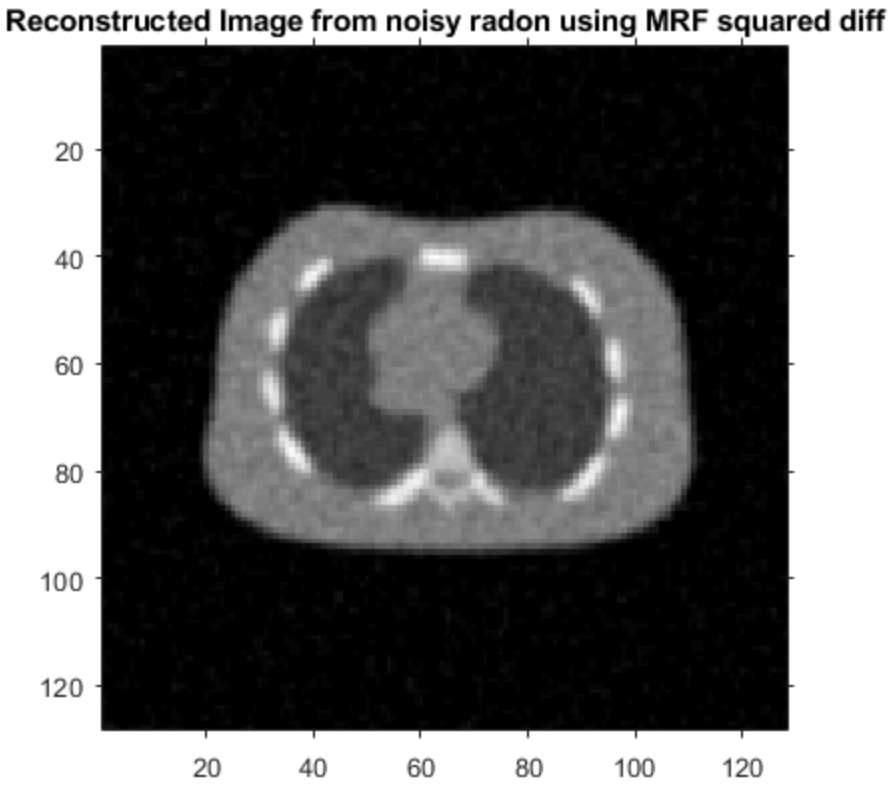


MRF Prior: squared difference

```
fprintf('MRF prior: Squared Difference\n');
alpha_opt = 56; gamma_opt = 1; eps = 1e-8; epochs = 100;
alpha = alpha_opt; gamma = gamma_opt;

[mrf_square, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @quad_grad, @quad_cost);

% RRMSE
```

```
error = sqrt(sum((mrf_square-img(:)).^2)/sum(img(:).^2));

% Optimal Parameters Testing
fprintf('alpha(a) = %.3f\n', alpha_opt);
fprintf('RRMSE(a) = %f\n', error);

alpha = 0.8 * alpha_opt; gamma = gamma_opt;
[mrf_square1, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @quad_grad, @quad_cost);
error = sqrt(sum((mrf_square1-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(0.8a) = %f\n', error);

alpha = 1.2 * alpha_opt; gamma = gamma_opt;
[mrf_square1, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @quad_grad, @quad_cost);
error = sqrt(sum((mrf_square1-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(1.2a) = %f\n', error);

% Plot image reconstructed
img_recon = reshape(mrf_square,[n,n]);
figure;
imshow(img_recon);
title('Reconstructed Image from noisy radon using MRF squared diff');
```

```
MRF prior: Squared Difference
alpha(a) = 56.000
RRMSE(a) = 0.136336
RRMSE(0.8a) = 0.137042
RRMSE(1.2a) = 0.136710
```

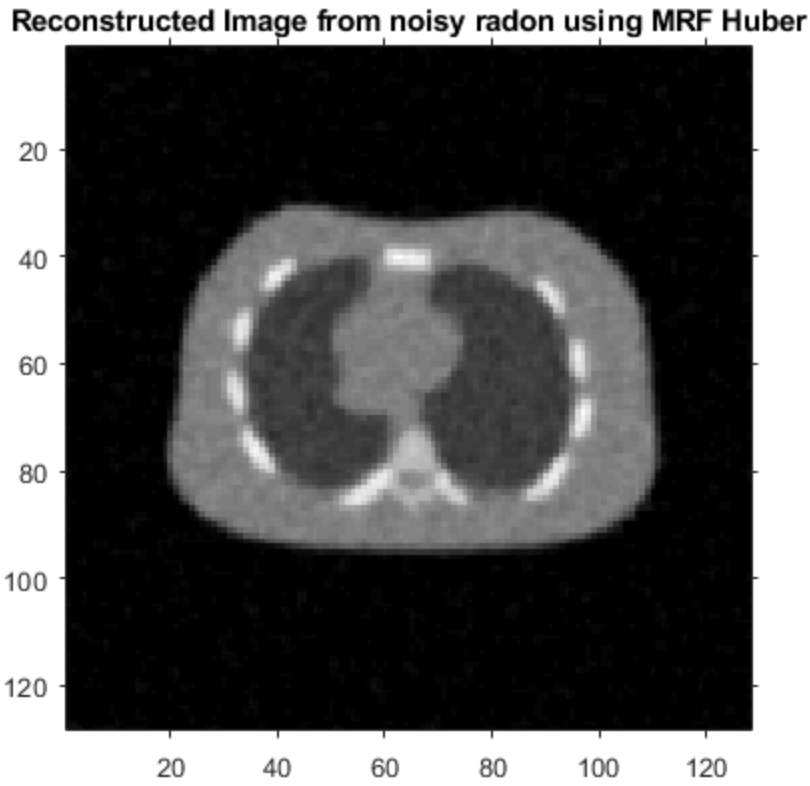


MRF Prior : Huber function

```
fprintf('MRF prior: Huber function\n');
alpha_opt = 180; gamma_opt = 0.042; eps = 1e-8; epochs = 100;
alpha = alpha_opt; gamma = gamma_opt;
[mrf_square_main, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @huber_grad, @huber_cost);

% RRMSE
error = sqrt(sum((mrf_square_main-img(:)).^2)/sum(img(:).^2));

% Optimal Parameters
fprintf('alpha(a) = %f, gamma(b) = %f\n', alpha_opt, gamma_opt);
fprintf('RRMSE(a, b) = %f\n', error);

alpha = 0.8 * alpha_opt; gamma = gamma_opt;
[mrf_square, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @huber_grad, @huber_cost);
error = sqrt(sum((mrf_square-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(0.8a, b) = %f\n', error);

alpha = alpha_opt; gamma = 0.8 * gamma_opt;
[mrf_square, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @huber_grad, @huber_cost);
error = sqrt(sum((mrf_square-img(:)).^2)/sum(img(:).^2));
```

```
fprintf('RRMSE(a, 0.8b) = %f\n', error);

alpha = 1.2 * alpha_opt; gamma = gamma_opt;
[mrf_square, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @huber_grad, @huber_cost);
error = sqrt(sum((mrf_square-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(1.2a, b) = %f\n', error);

alpha = alpha_opt; gamma = 1.2 * gamma_opt;
[mrf_square, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @huber_grad, @huber_cost);
error = sqrt(sum((mrf_square-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(a, 1.2b) = %f\n', error);

% Plot image reconstructed
img_recon = reshape(mrf_square_main,[n,n]);
figure;
imshow(img_recon);
title('Reconstructed Image from noisy radon using MRF Huber');
```

```
MRF prior: Huber function
alpha(a) = 180.000000, gamma(b) = 0.042000
RRMSE(a, b) = 0.126001
RRMSE(0.8a, b) = 0.126321
RRMSE(a, 0.8b) = 0.126391
RRMSE(1.2a, b) = 0.127101
RRMSE(a, 1.2b) = 0.126420
```

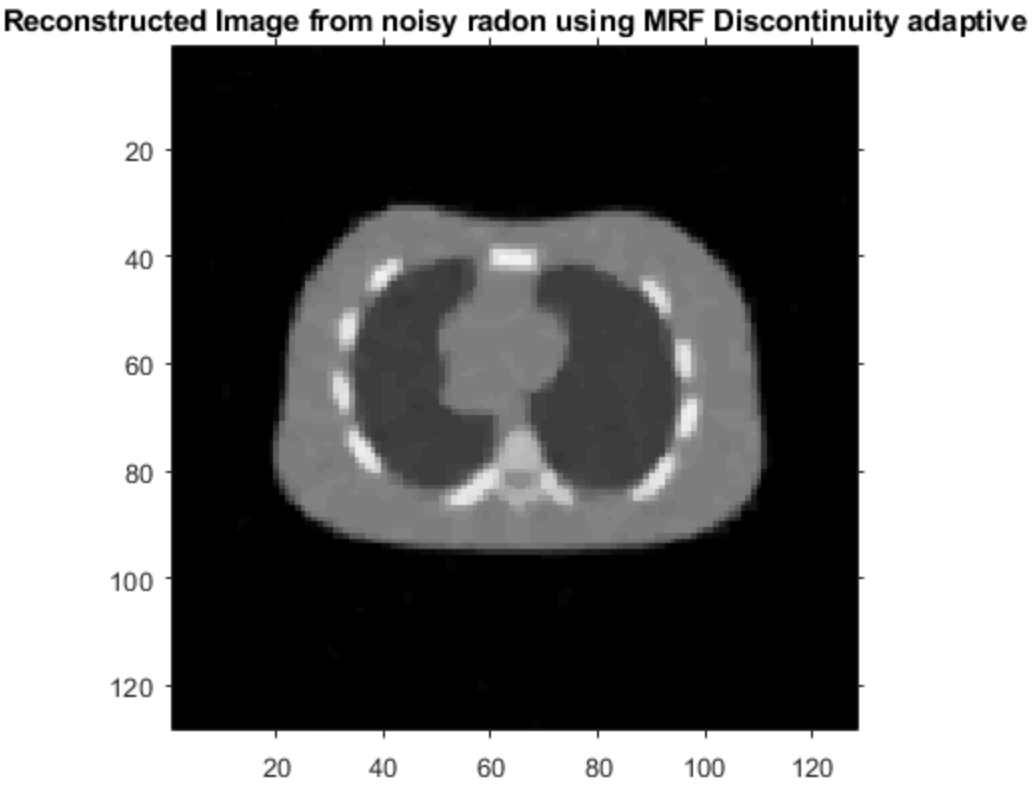


MRF Prior : Discontinuity-adaptive Log function

```
fprintf('MRF prior: Discontinuity-adaptive Log function\n');
alpha_opt = 6014; gamma_opt = 0.0034; eps = 1e-8; epochs = 100;
alpha = alpha_opt; gamma = gamma_opt;
[mrf_square_main, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @log_grad, @log_cost);

% RRMSE
error = sqrt(sum((mrf_square_main-img(:)).^2)/sum(img(:).^2));

% Optimal Parameters
fprintf('alpha(a) = %f, gamma(b) = %f\n', alpha_opt, gamma_opt);
fprintf('RRMSE(a, b) = %f\n', error);

alpha = 0.8 * alpha_opt; gamma = gamma_opt;
[mrf_square, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @log_grad, @log_cost);
error = sqrt(sum((mrf_square-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(0.8a, b) = %f\n', error);

alpha = alpha_opt; gamma = 0.8 * gamma_opt;
```

```
[mrf_square, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @log_grad, @log_cost);
error = sqrt(sum((mrf_square-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(a, 0.8b) = %f\n', error);

alpha = 1.2 * alpha_opt; gamma = gamma_opt;
[mrf_square, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @log_grad, @log_cost);
error = sqrt(sum((mrf_square-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(1.2a, b) = %f\n', error);

alpha = alpha_opt; gamma = 1.2 * gamma_opt;
[mrf_square, ~] = descent(A, noised_radon(:), eps, alpha, gamma,
 epochs, @log_grad, @log_cost);
error = sqrt(sum((mrf_square-img(:)).^2)/sum(img(:).^2));
fprintf('RRMSE(a, 1.2b) = %f\n', error);

% Plot image reconstructed
img_recon = reshape(mrf_square_main,[n,n]);
figure;
imshow(img_recon);
title('Reconstructed Image from noisy radon using MRF Discontinuity
 adaptive');
```

```
MRF prior: Discontinuity-adaptive Log function
alpha(a) = 6014.000000, gamma(b) = 0.003400
RRMSE(a, b) = 0.095445
RRMSE(0.8a, b) = 0.096145
RRMSE(a, 0.8b) = 0.095569
RRMSE(1.2a, b) = 0.094951
RRMSE(a, 1.2b) = 0.096164
```



Published with MATLAB® R2019a