Strategy for RGB denoising

We run the algorithm used in Q1 for all the 3 channels of the RGB image separately. We keep the hyper-parameters alpha and gamma same for each of the channels, i.e alpha_red = alpha_green = alpha_blue and gamma_red = gamma_green = gamma_blue.

This is equivalent to having the objective cost function as the sum of the cost functions for all the individual layers. Because while taking the derivative of the cost funtion with repect to any pixel, only those pixels would remain which are in the same channel, as the MRF prior considers only the neighbours in the same channel. We keep the cost function as mentioned above and not have all different alphas and gammas because by this we can capture to some extent the fact that the layers are not totally independent. And this is done by having all their contributions in the cost function.

If the alphas and gammas were independent of each other, i.e we would do tuning separately for each layer, then that would imply that the RGB channels are totally unrelated, which is not what we want.

Now, with the method used, we can also keep different hyper-parameters for each layer, in case we want to consider different amounts of noise in each layer.

Contents

Prepping the script

```
clc; clear; close all;
addpath('../functions/');
actual = abs(double(imread('../../data/histology_noiseless.png'))/255);
```

Reading the data

```
fprintf('Analysis for RGB image\n');
img = abs(double(imread('../../data/histology_noisy.png'))/255);
```

```
Analysis for RGB image
```

MRF prior: Quadratic function

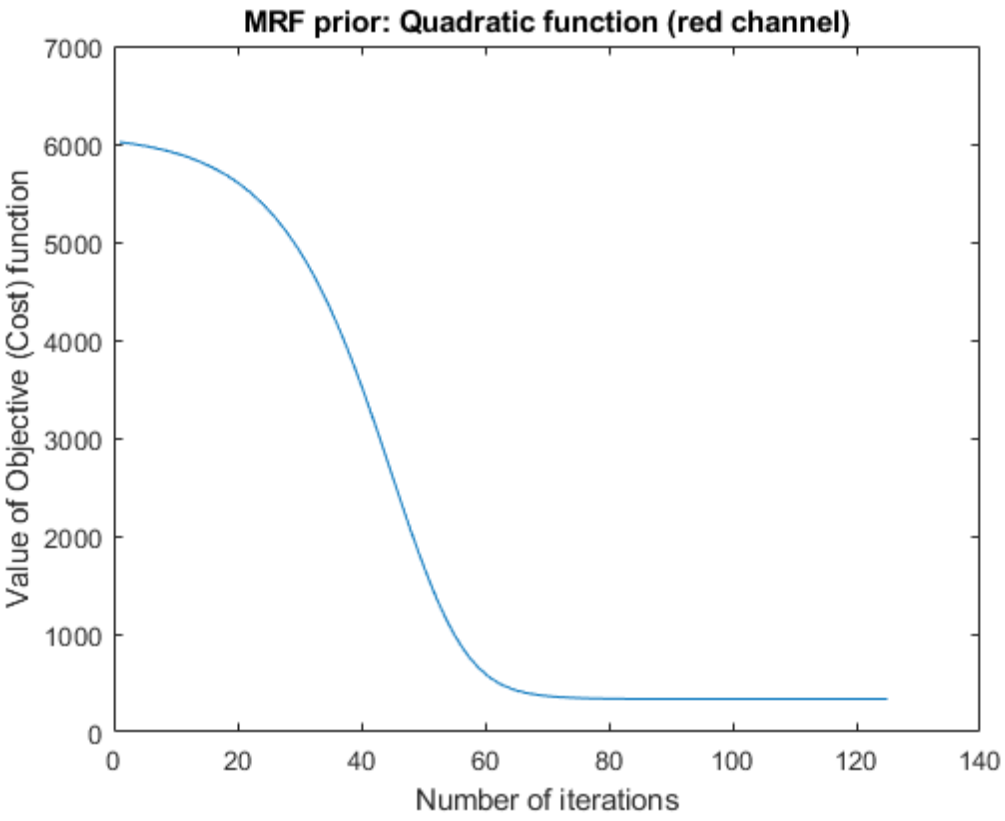
```
fprintf('MRF prior: Quadratic function\n');
alpha_opt = 0.705; gamma_opt = 1; eps = 1e-8; epochs = 1e9;

alpha = alpha_opt; gamma = gamma_opt;
[noisy1, costs1] = descent(img(:,:,1), eps, alpha, gamma, epochs, @quad_grad, @quad_cost);
[noisy2, costs2] = descent(img(:,:,2), eps, alpha, gamma, epochs, @quad_grad, @quad_cost);
[noisy3, costs3] = descent(img(:,:,3), eps, alpha, gamma, epochs, @quad_grad, @quad_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
quad_denoised = abs(noisy);

% RRMSE
error = rrmse(actual, img);
fprintf('RRMSE(noiseless, noisy) : %f\n',error);

% Optimal Parameters Testing
fprintf('alpha(a) = %f\n', alpha_opt);
error = rrmse(actual, noisy);
fprintf('RRMSE(a) = %f\n', error);

alpha = 0.8 * alpha_opt; gamma = gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @quad_grad, @quad_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @quad_grad, @quad_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @quad_grad, @quad_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(0.8a) = %f\n', error);

alpha = 1.2 * alpha_opt; gamma = gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @quad_grad, @quad_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @quad_grad, @quad_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @quad_grad, @quad_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(1.2a) = %f\n', error);
```

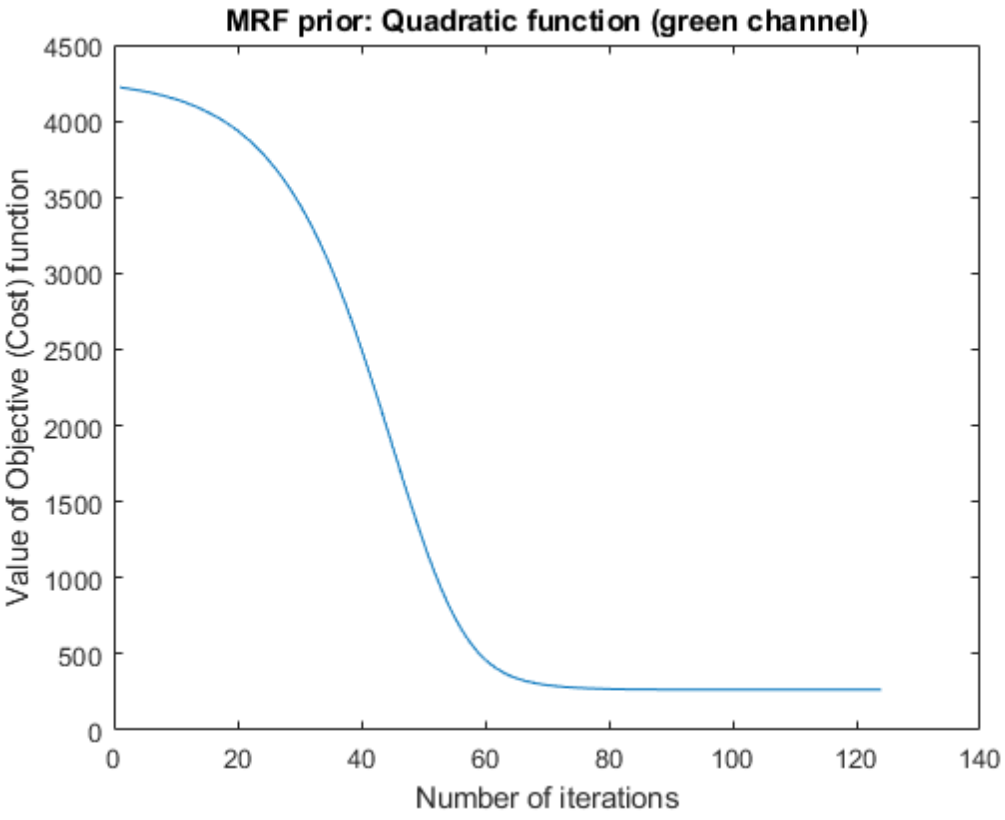
```
MRF prior: Quadratic function
RRMSE(noiseless, noisy) : 0.199383
alpha(a) = 0.705000
RRMSE(a) = 0.053565
RRMSE(0.8a) = 0.059414
RRMSE(1.2a) = 0.058610
```

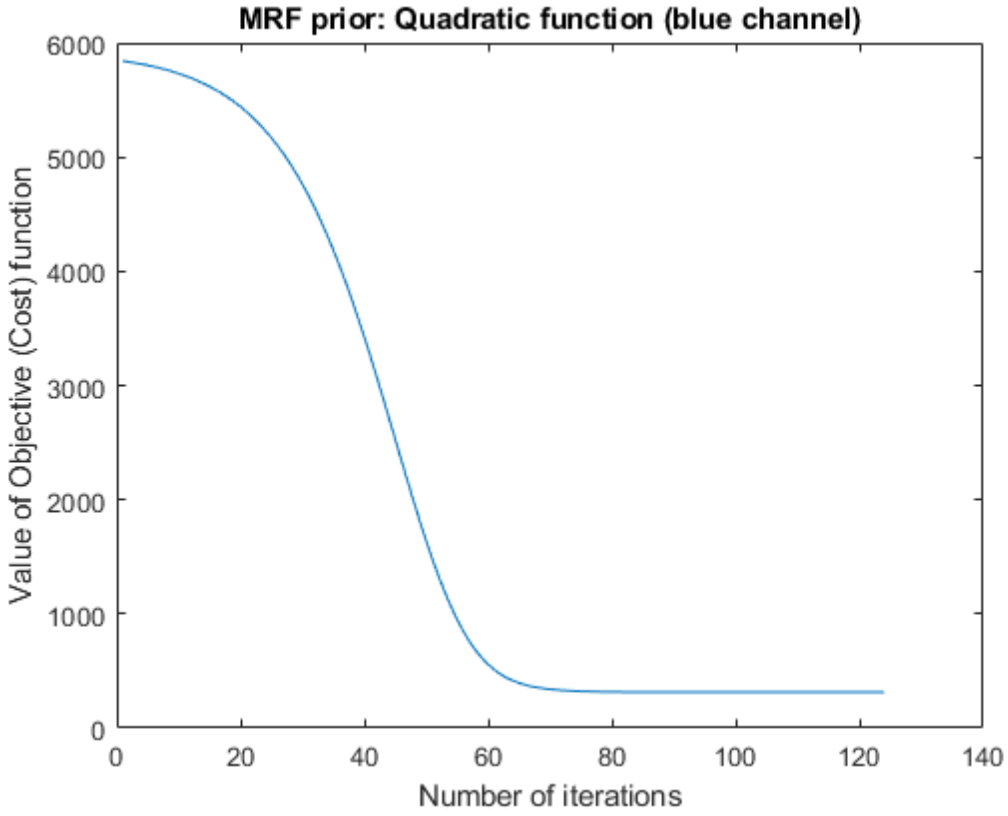
Objective function plot for Quadratic Function MRF

```
figure;
plot(costs1);
xlabel('Number of iterations')
ylabel('Value of Objective (Cost) function');
title('MRF prior: Quadratic function (red channel)');

figure;
plot(costs2);
xlabel('Number of iterations')
ylabel('Value of Objective (Cost) function');
title('MRF prior: Quadratic function (green channel)');

figure;
plot(costs3);
xlabel('Number of iterations')
ylabel('Value of Objective (Cost) function');
title('MRF prior: Quadratic function (blue channel)');
```

```
magnification = 200;

%-------------------------%
```



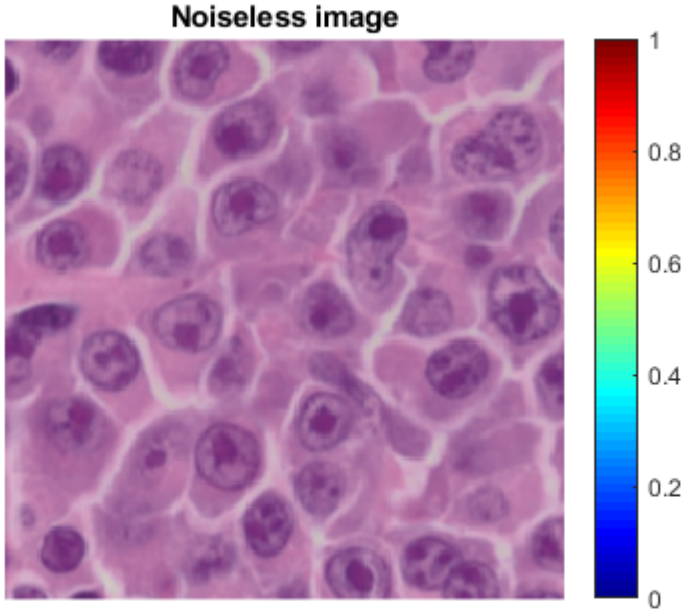


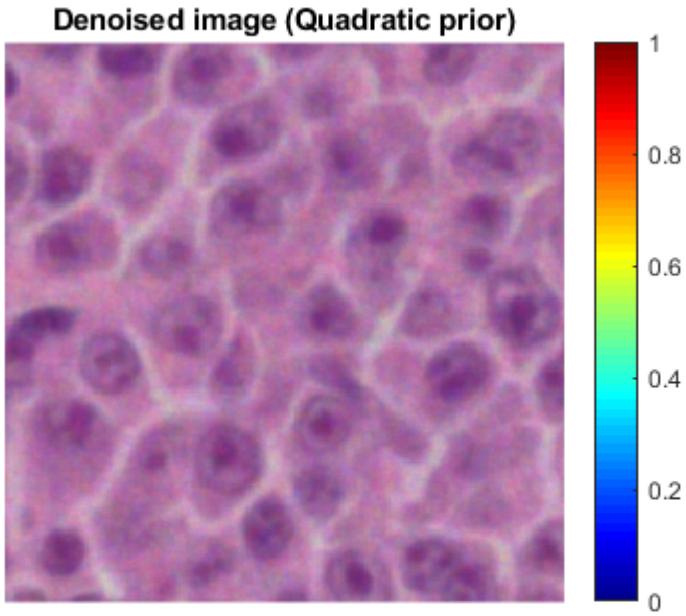


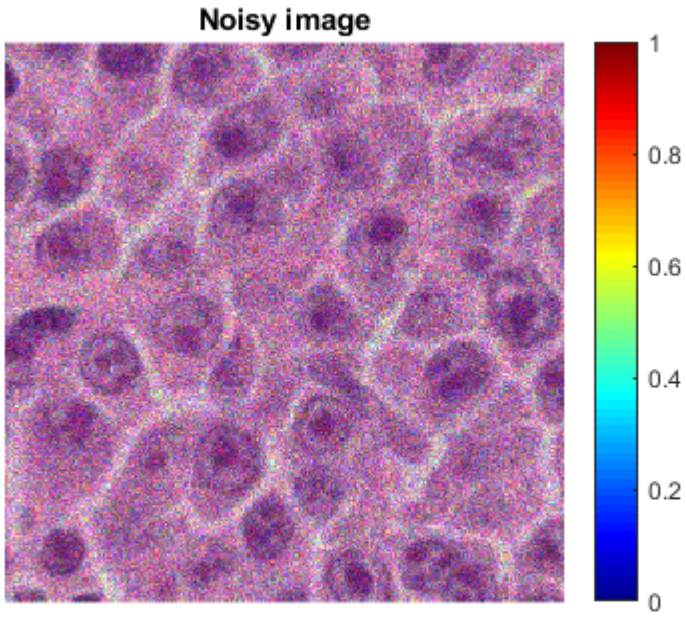
Colormap plot for Quadratic Prior

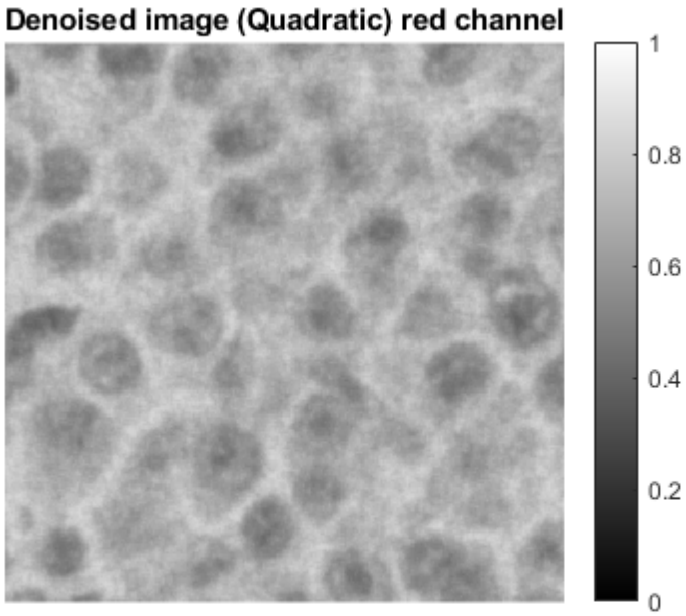
```
figure;
colormap(jet)
imshow(actual, 'InitialMagnification', magnification);
title('Noiseless image');
colorbar

figure;
colormap(jet)
imshow(quad_denoised, 'InitialMagnification', magnification);
title('Denoised image (Quadratic prior)');
colorbar

figure;
colormap(jet)
imshow(img, 'InitialMagnification', magnification);
title('Noisy image');
colorbar

figure;
colormap(jet)
imshow(quad_denoised(:,:,1), 'InitialMagnification', magnification);
title('Denoised image (Quadratic) red channel');
colorbar

figure;
colormap(jet)
imshow(quad_denoised(:,:,2), 'InitialMagnification', magnification);
title('Denoised image (Quadratic) green channel');
colorbar

figure;
colormap(jet)
imshow(quad_denoised(:,:,3), 'InitialMagnification', magnification);
title('Denoised image (Quadratic) blue channel');
```

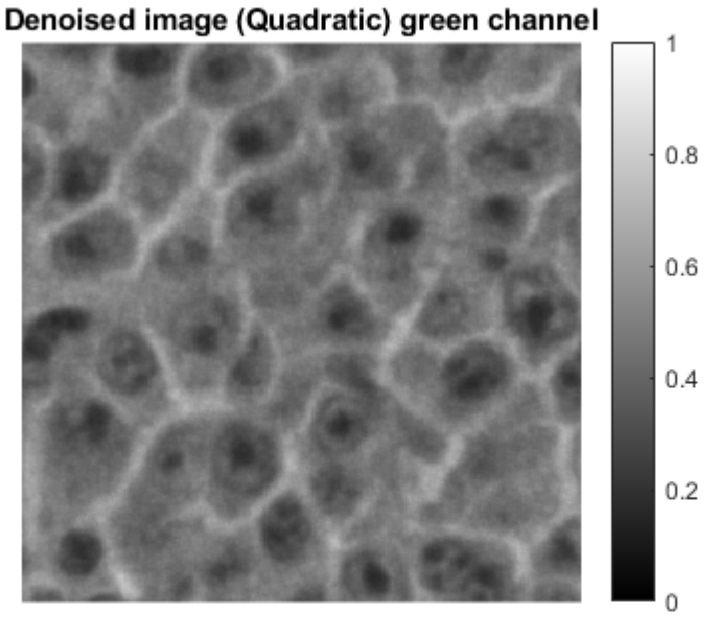
```
colorbar
%--------------------%
```

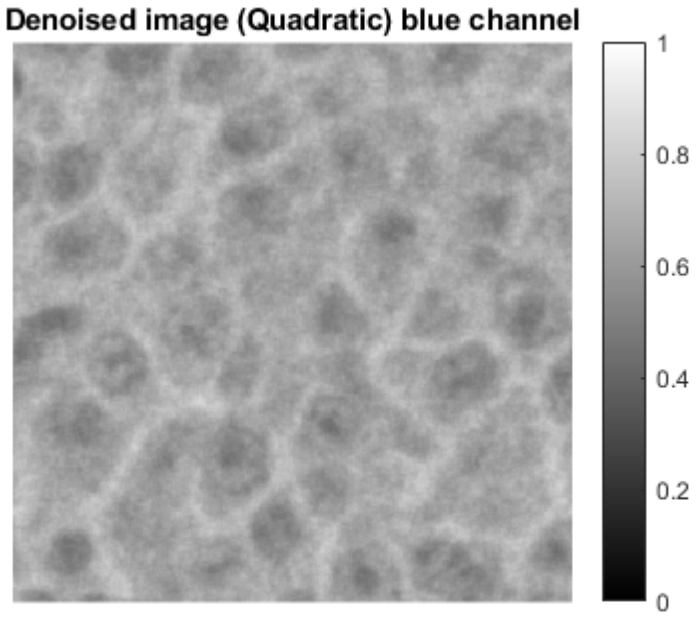

Noiseless image
1
0.8
0.6
0.4
0.2
0


Denoised image (Quadratic prior)
1
0.8
0.6
0.4
0.2
0


Noisy image
1
0.8
0.6
0.4
0.2
0


Denoised image (Quadratic) red channel
1
0.8
0.6
0.4
0.2
0





MRF prior: Discontinuity-adaptive Huber function

```
fprintf('MRF prior: Discontinuity-adaptive Huber function\n');
alpha_opt = 0.8; gamma_opt = 0.064; eps = 1e-8; epochs = 1e9;

alpha = alpha_opt; gamma = gamma_opt;
[noisy1, costs1] = descent(img(:,:,1), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
[noisy2, costs2] = descent(img(:,:,2), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
```

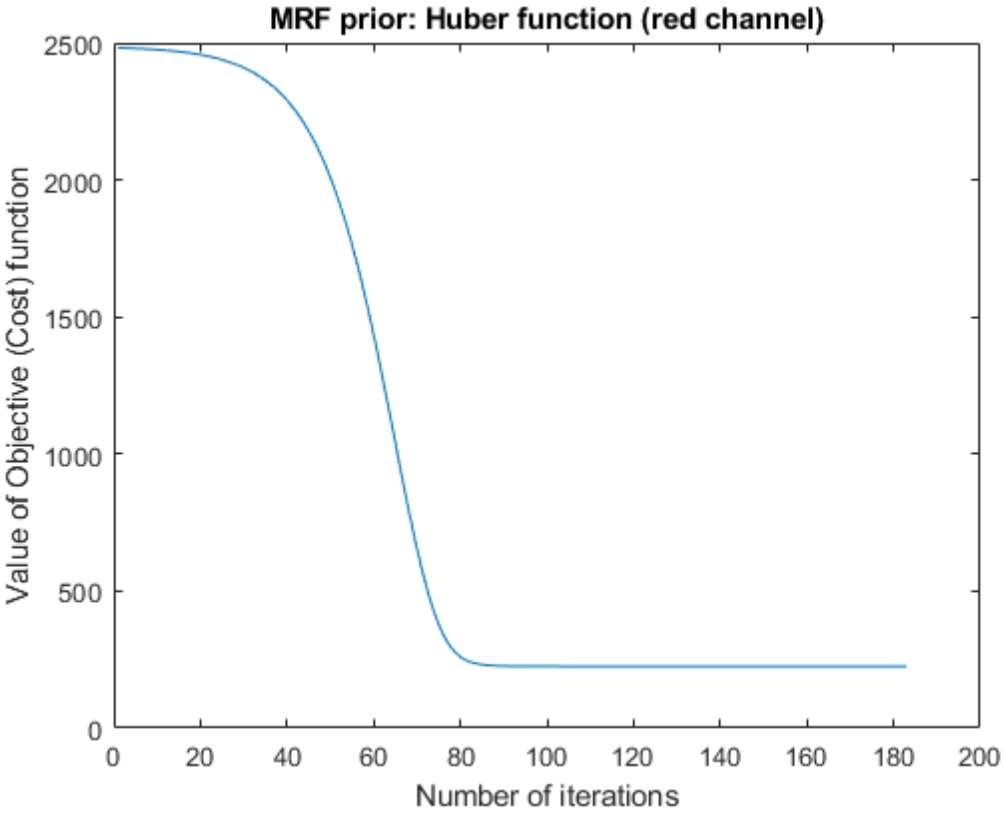
```
[noisy3, costs3] = descent(img(:,:,3), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
cdim = min(min(size(costs1,2),size(costs2,2)),size(costs3,2));
costs = (costs1(:,1:cdim)+costs2(:,1:cdim)+costs3(:,1:cdim))/3;
noisy = cat(3,noisy1,noisy2,noisy3);
huber_denoised = abs(noisy);

% RRMSE
error = rrmse(actual, img);
fprintf('RRMSE(noiseless, noisy) : %f\n',error);

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

alpha = 0.8 * alpha_opt; gamma = gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(0.8a, b) = %f\n', error);

alpha = alpha_opt; gamma = 0.8 * gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(a, 0.8b) = %f\n', error);

alpha = 1.2 * alpha_opt; gamma = gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(1.2a, b) = %f\n', error);

alpha = alpha_opt; gamma = 1.2 * gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @huber_grad, @huber_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(a, 1.2b) = %f\n', error);
```

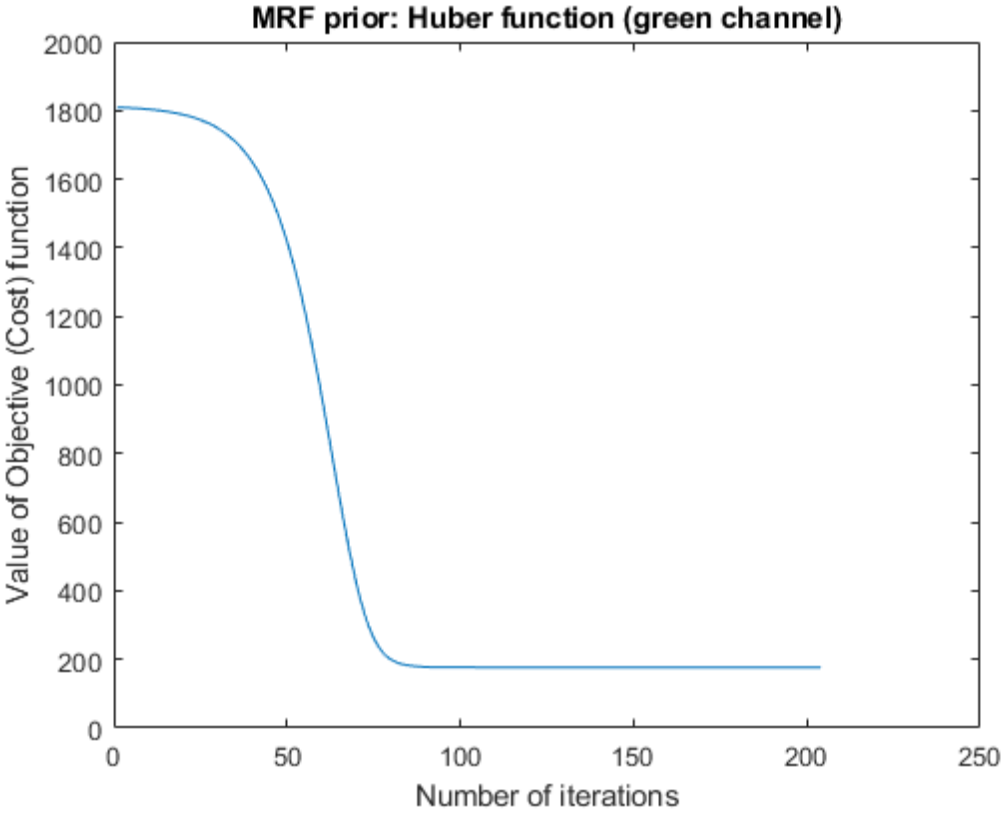
```
MRF prior: Discontinuity-adaptive Huber function
RRMSE(noiseless, noisy) : 0.199383
alpha(a) = 0.800000, gamma(b) = 0.064000
RRMSE(a, b) = 0.053964
RRMSE(0.8a, b) = 0.075131
RRMSE(a, 0.8b) = 0.054213
RRMSE(1.2a, b) = 0.072406
RRMSE(a, 1.2b) = 0.054090
```

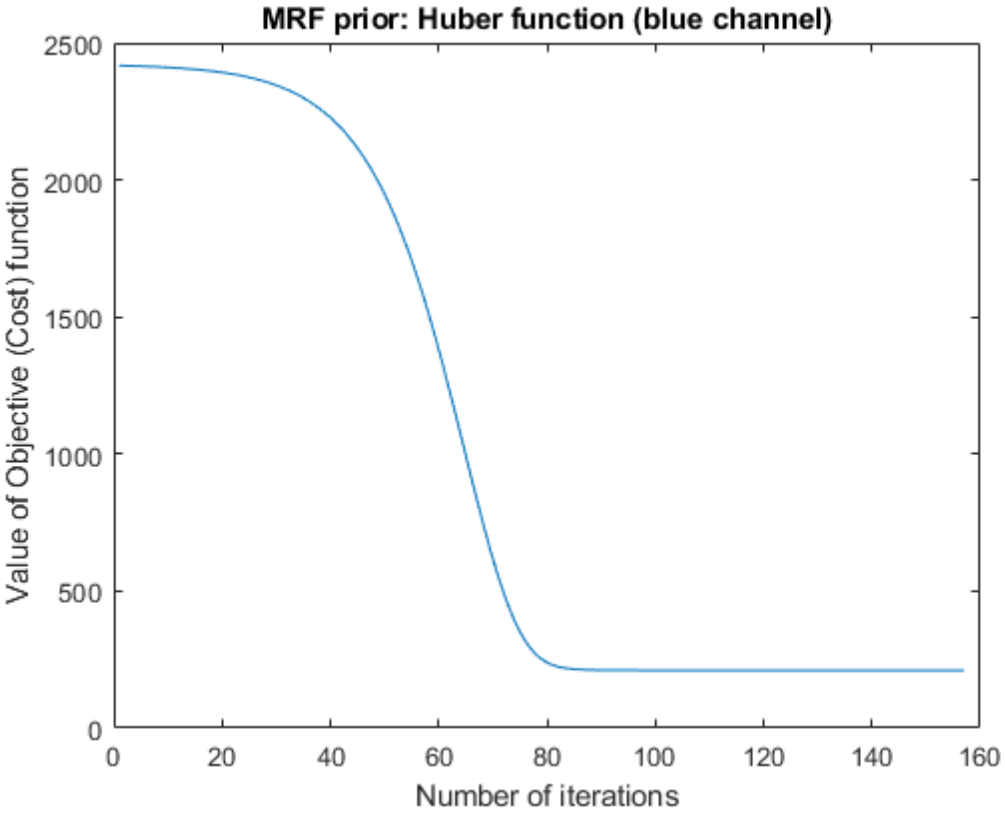
Objective function plot for Huber Function MRF

```
figure;
plot(costs1);
xlabel('Number of iterations')
ylabel('Value of Objective (Cost) function');
```

```
title('MRF prior: Huber function (red channel)');

figure;
plot(costs2);
xlabel('Number of iterations')
ylabel('Value of Objective (Cost) function');
title('MRF prior: Huber function (green channel)');

figure;
plot(costs3);
xlabel('Number of iterations')
ylabel('Value of Objective (Cost) function');
title('MRF prior: Huber function (blue channel)');

%-------------------%
```


MRF prior: Huber function (red channel)
Value of Objective (Cost) function
Number of iterations
0
500
1000
1500
2000
2500
0
20
40
60
80
100
120
140
160
180
200

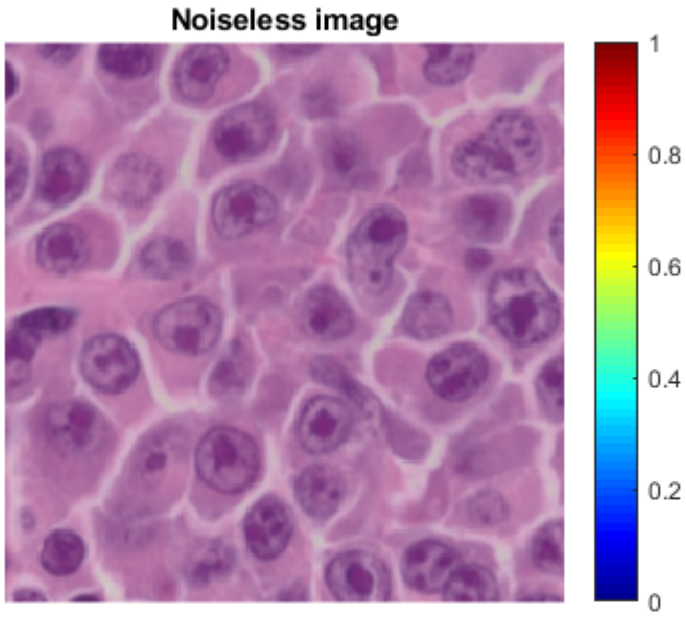

MRF prior: Huber function (green channel)
Value of Objective (Cost) function
Number of iterations
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0
50
100
150
200
250

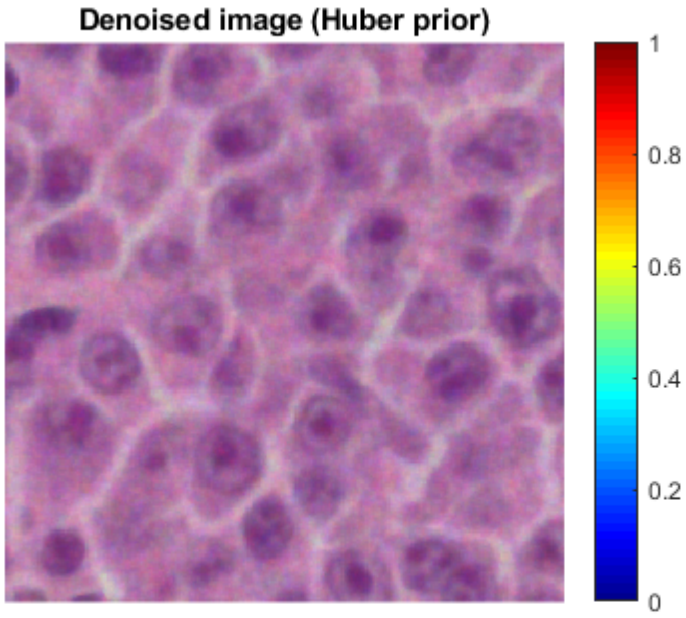


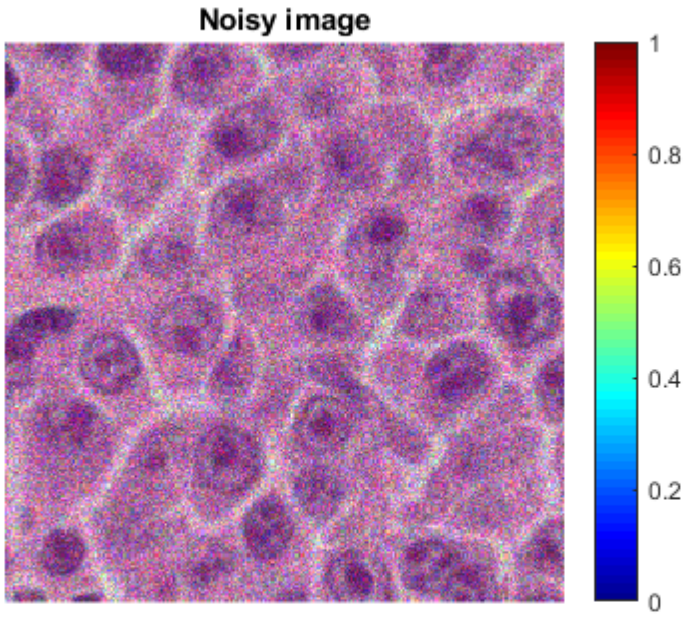
Colormap plot for Huber Prior

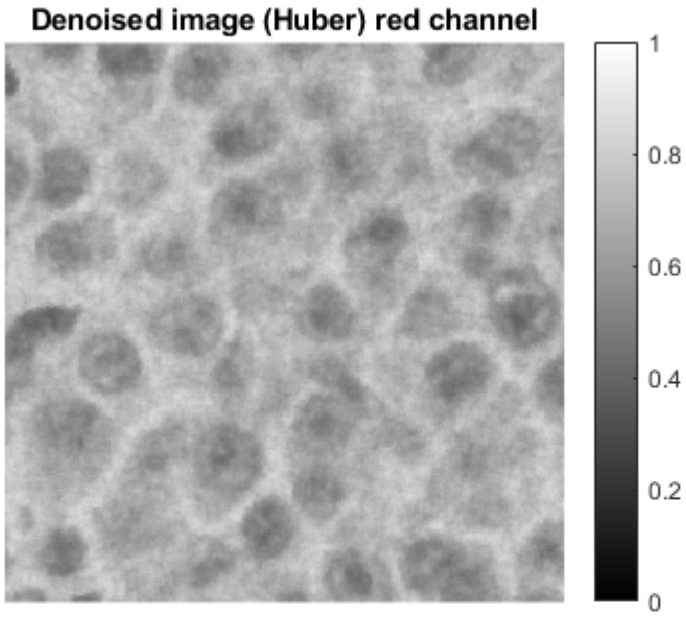
```
figure;
colormap(jet)
imshow(actual, 'InitialMagnification', magnification);
title('Noiseless image');
colorbar

figure;
colormap(jet)
imshow(huber_denoised, 'InitialMagnification', magnification);
title('Denoised image (Huber prior)');
colorbar

figure;
colormap(jet)
imshow(img, 'InitialMagnification', magnification);
title('Noisy image');
colorbar

figure;
colormap(jet)
imshow(huber_denoised(:,:,1), 'InitialMagnification', magnification);
title('Denoised image (Huber) red channel');
colorbar

figure;
colormap(jet)
imshow(huber_denoised(:,:,2), 'InitialMagnification', magnification);
title('Denoised image (Huber) green channel');
colorbar

figure;
colormap(jet)
imshow(huber_denoised(:,:,3), 'InitialMagnification', magnification);
title('Denoised image (Huber) blue channel');
colorbar
```

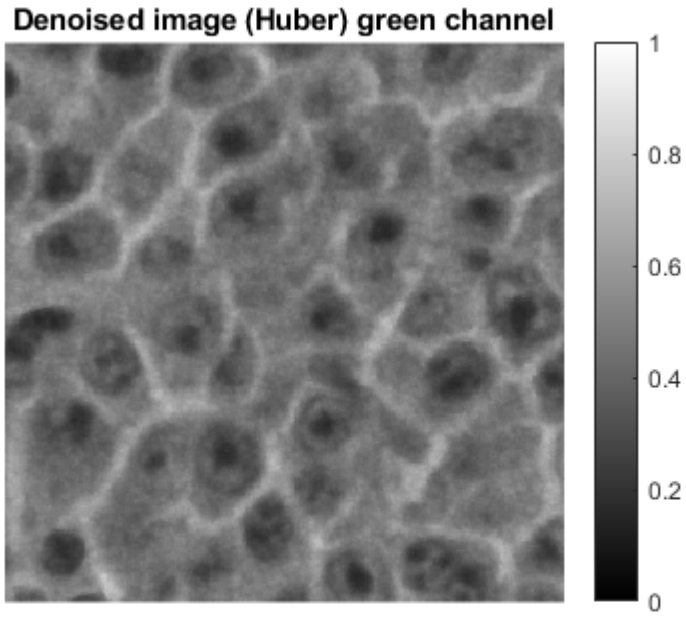
```
%--------------------%
```

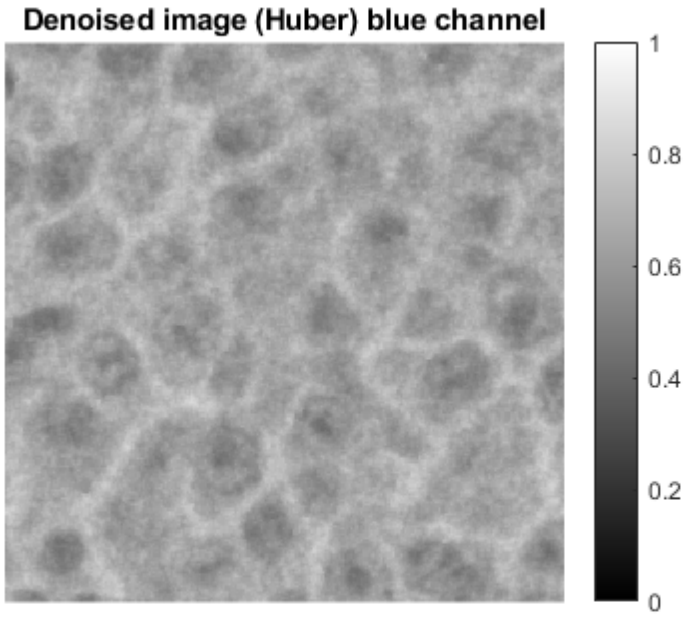

Noiseless image
1
0.8
0.6
0.4
0.2
0


Denoised image (Huber prior)
1
0.8
0.6
0.4
0.2
0


Noisy image
1
0.8
0.6
0.4
0.2
0


Denoised image (Huber) red channel
1
0.8
0.6
0.4
0.2
0





MRF prior: Discontinuity-adaptive Log function

```
fprintf('MRF prior: Discontinuity-adaptive Log function\n');
alpha_opt = 0.80; gamma_opt = 20.1; eps = 1e-8; epochs = 1e9;

alpha = alpha_opt; gamma = gamma_opt;
[noisy1, costs1] = descent(img(:,:,1), eps, alpha, gamma, epochs, @log_grad, @log_cost);
[noisy2, costs2] = descent(img(:,:,2), eps, alpha, gamma, epochs, @log_grad, @log_cost);
```

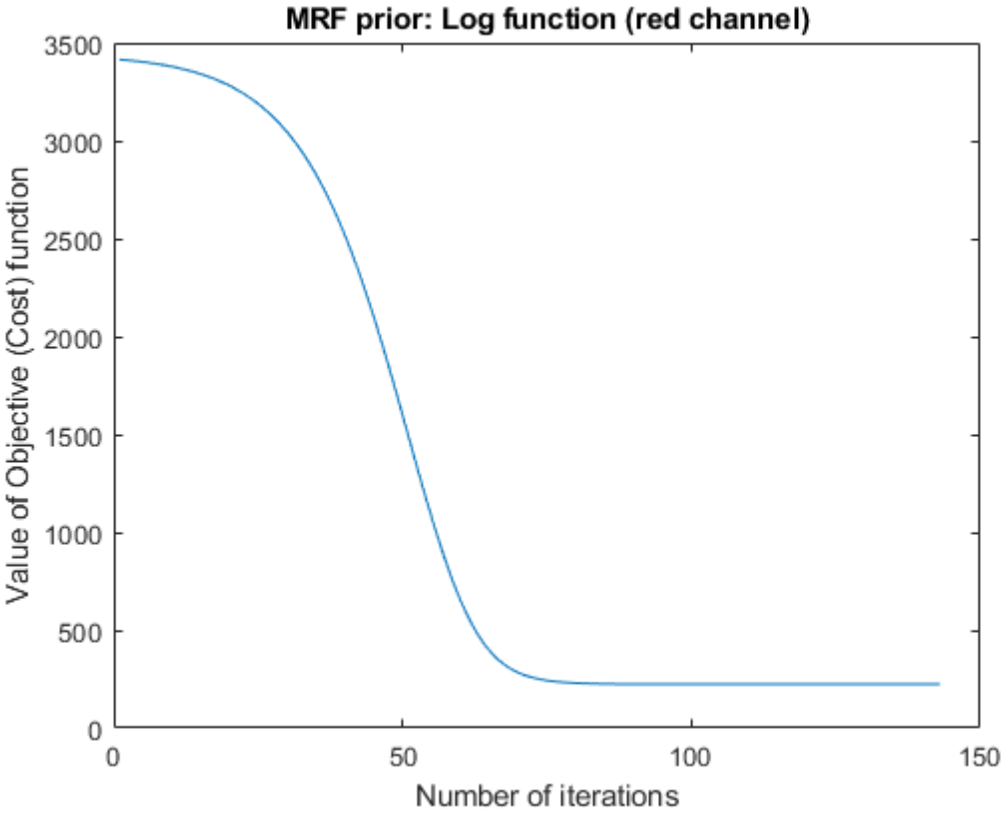
```
[noisy3, costs3] = descent(img(:,:,3), eps, alpha, gamma, epochs, @log_grad, @log_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
log_denoised = abs(noisy);

% RRMSE
error = rrmse(actual, img);
fprintf('RRMSE(noiseless, noisy) : %f\n',error);

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

alpha = 0.8 * alpha_opt; gamma = gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @log_grad, @log_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @log_grad, @log_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @log_grad, @log_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(0.8a, b) = %f\n', error);

alpha = alpha_opt; gamma = 0.8 * gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @log_grad, @log_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @log_grad, @log_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @log_grad, @log_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(a, 0.8b) = %f\n', error);

alpha = 1.2 * alpha_opt; gamma = gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @log_grad, @log_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @log_grad, @log_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @log_grad, @log_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(1.2a, b) = %f\n', error);

alpha = alpha_opt; gamma = 1.2 * gamma_opt;
[noisy1, ~] = descent(img(:,:,1), eps, alpha, gamma, epochs, @log_grad, @log_cost);
[noisy2, ~] = descent(img(:,:,2), eps, alpha, gamma, epochs, @log_grad, @log_cost);
[noisy3, ~] = descent(img(:,:,3), eps, alpha, gamma, epochs, @log_grad, @log_cost);
noisy = cat(3,noisy1,noisy2,noisy3);
error = rrmse(actual, noisy);
fprintf('RRMSE(a, 1.2b) = %f\n', error);

%-------------------%
```

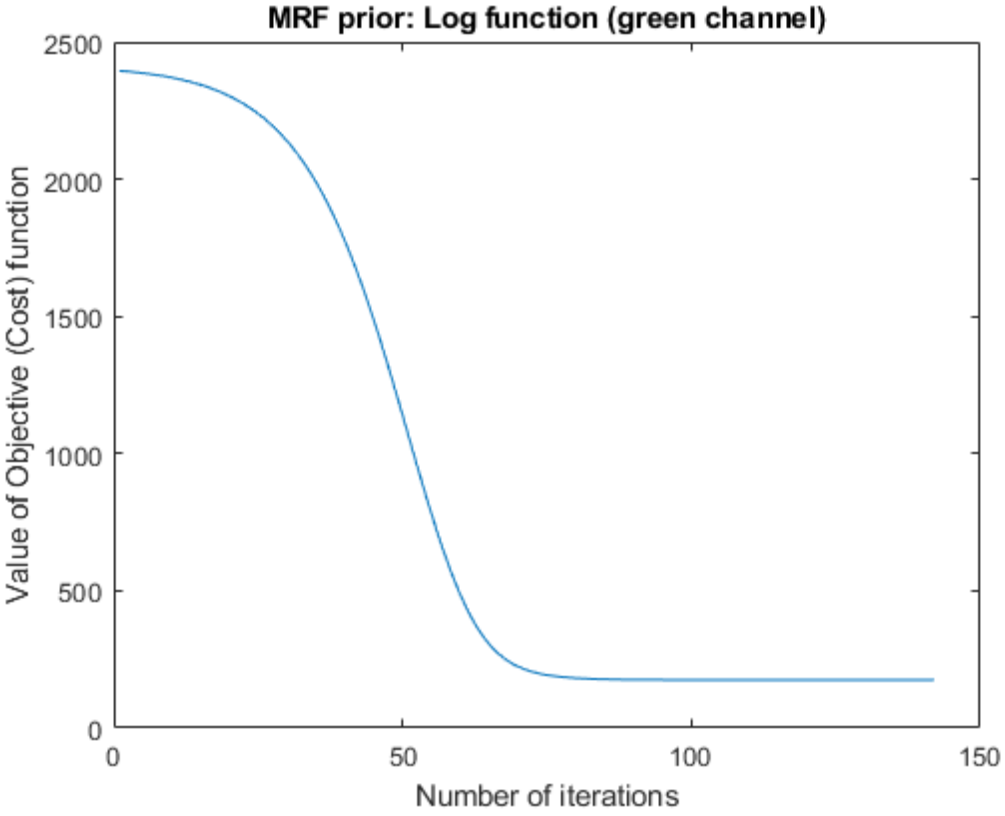
```
MRF prior: Discontinuity-adaptive Log function
RRMSE(noiseless, noisy) : 0.199383
alpha(a) = 0.800000, gamma(b) = 20.100000
RRMSE(a, b) = 0.054326
RRMSE(0.8a, b) = 0.067474
RRMSE(a, 0.8b) = 0.054328
RRMSE(1.2a, b) = 0.072390
RRMSE(a, 1.2b) = 0.054324
```

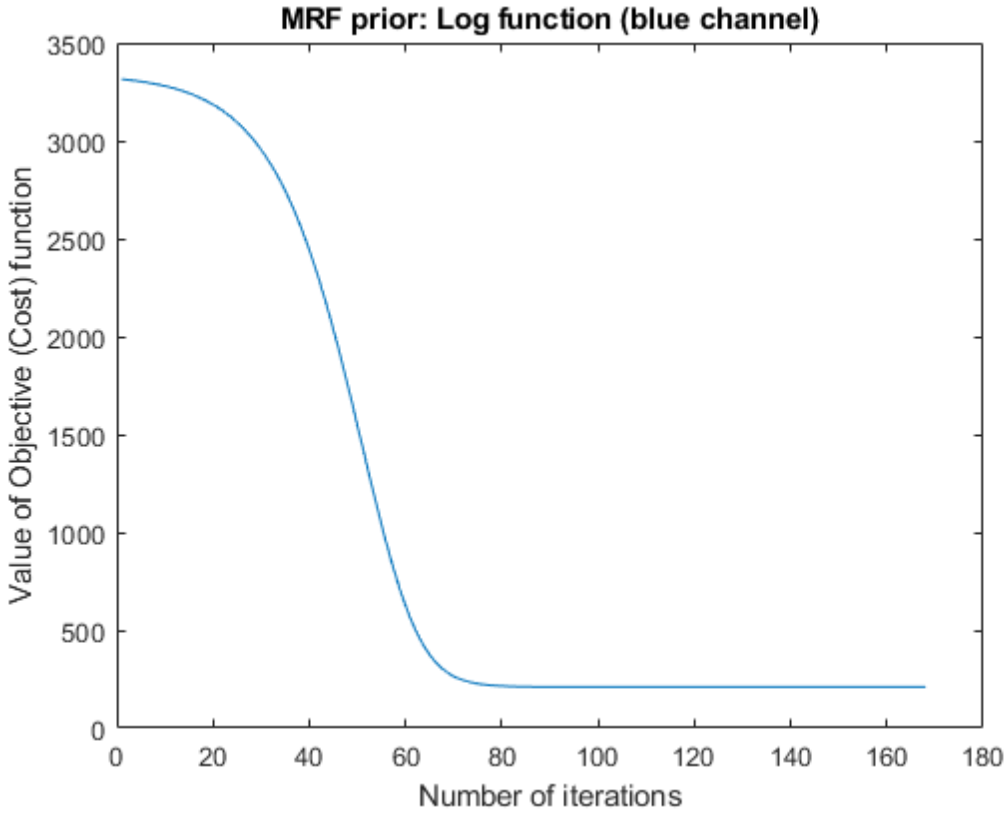
Objective function plot for Log Function MRF

```
figure;
plot(costs1);
xlabel('Number of iterations')
ylabel('Value of Objective (Cost) function');
```

```
title('MRF prior: Log function (red channel)');

figure;
plot(costs2);
xlabel('Number of iterations')
ylabel('Value of Objective (Cost) function');
title('MRF prior: Log function (green channel)');

figure;
plot(costs3);
xlabel('Number of iterations')
ylabel('Value of Objective (Cost) function');
title('MRF prior: Log function (blue channel)');

%-------------------%
```


MRF prior: Log function (red channel)
Value of Objective (Cost) function
Number of iterations
0
500
1000
1500
2000
2500
3000
3500
0
50
100
150

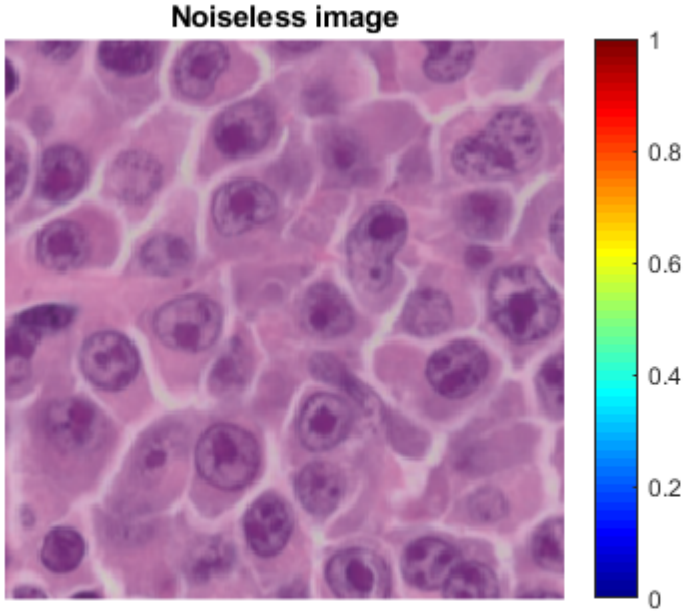

MRF prior: Log function (green channel)
Value of Objective (Cost) function
Number of iterations
0
500
1000
1500
2000
2500
0
50
100
150

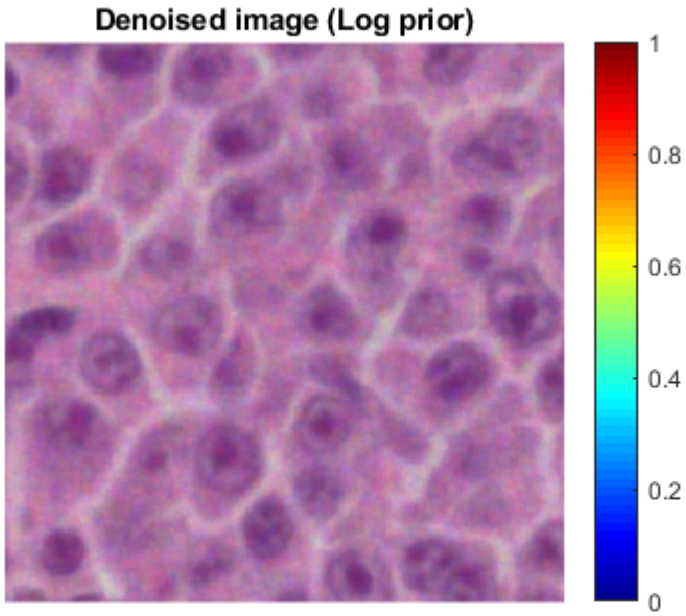


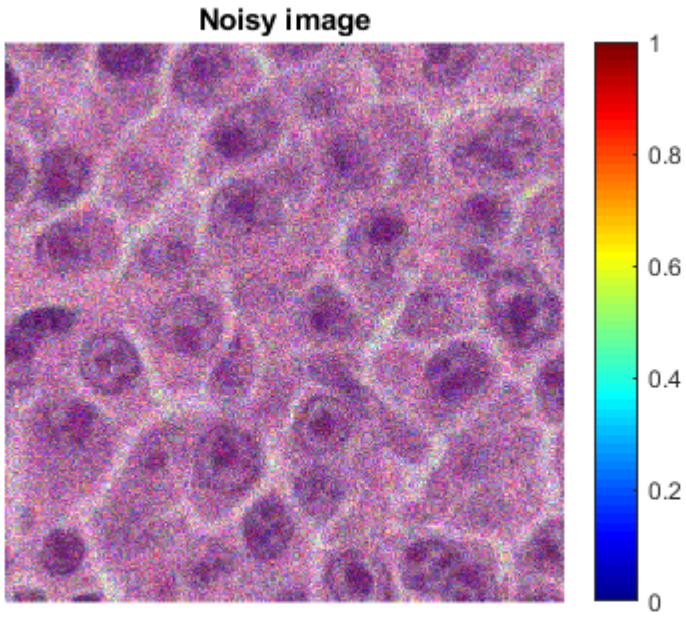
Colormap plot for Log Prior

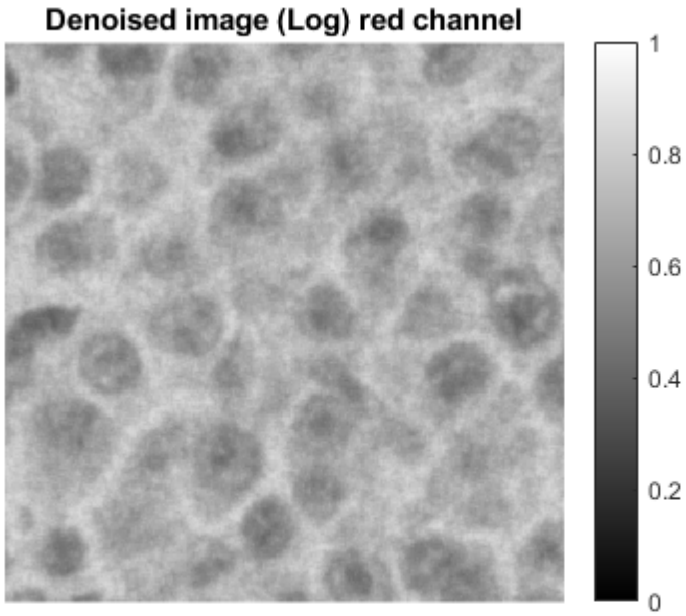
```
figure;
colormap(jet)
imshow(actual, 'InitialMagnification', magnification);
title('Noiseless image');
colorbar

figure;
colormap(jet)
imshow(log_denoised, 'InitialMagnification', magnification);
title('Denoised image (Log prior)');
colorbar

figure;
colormap(jet)
imshow(img, 'InitialMagnification', magnification);
title('Noisy image');
colorbar

figure;
colormap(jet)
imshow(log_denoised(:,:,1), 'InitialMagnification', magnification);
title('Denoised image (Log) red channel');
colorbar

figure;
colormap(jet)
imshow(log_denoised(:,:,2), 'InitialMagnification', magnification);
title('Denoised image (Log) green channel');
colorbar

figure;
colormap(jet)
imshow(log_denoised(:,:,3), 'InitialMagnification', magnification);
title('Denoised image (Log) blue channel');
colorbar
```

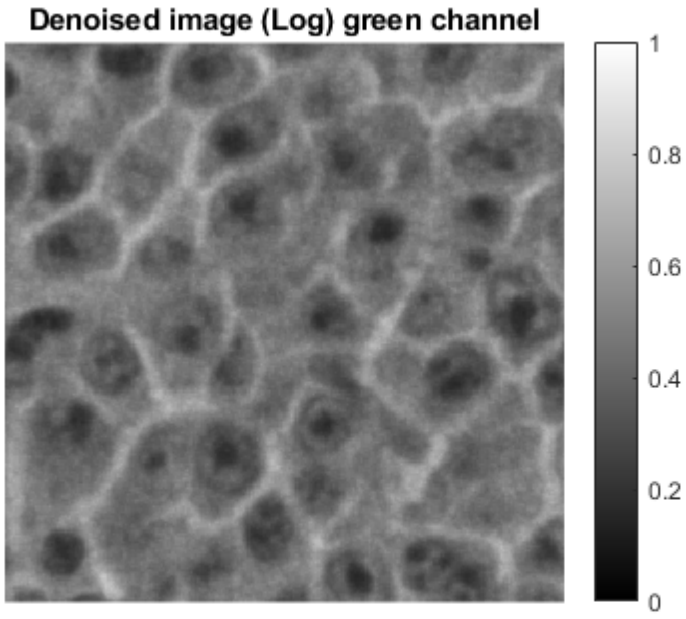
```
%----------%
```

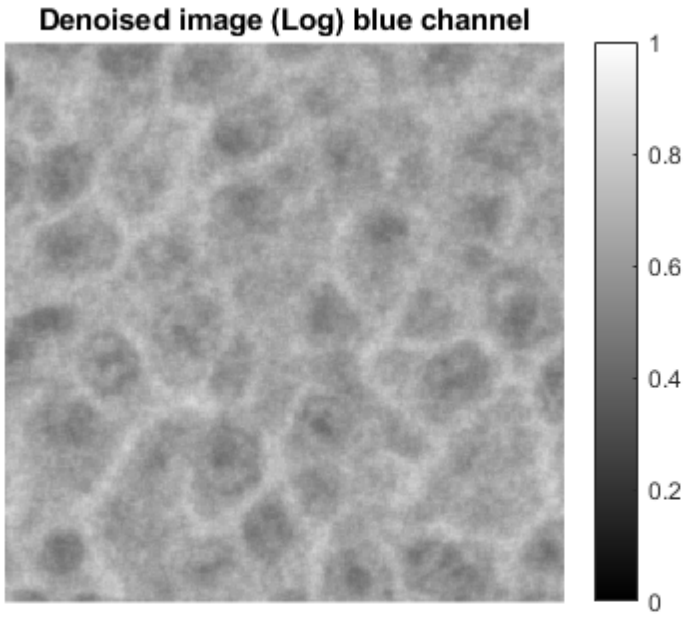

Noiseless image
1
0.8
0.6
0.4
0.2
0


Denoised image (Log prior)
1
0.8
0.6
0.4
0.2
0


Noisy image
1
0.8
0.6
0.4
0.2
0


Denoised image (Log) red channel
1
0.8
0.6
0.4
0.2
0





Published with MATLAB® R2019a