Assignment 1

Sankalp Parashar and Utkarsh Ranjan

Spring 2022

Contents

Question 1

Noise Model : Gaussian

Results

RRMSE between Noisy and Noiseless image : 0.2986

Quadratic MRF Prior

- α* **:** 0.055
- **RRMSE(**α***) :** 0.2812
- **RRMSE(1.2** α***) :** 0.2817
- **RRMSE(0.8** α***) :** 0.2815

Huber MRF Prior

- α* **:** 0.99
- γ* **:** 0.001
- **RRMSE**α***,**γ***) :** 0.236
- **RRMSE(1.2** α***,** γ***) :** 1.577e+29
- **RRMSE(0.8** α***,** γ***) :** 0.2796
- **RRMSE(**α***, 1.2** γ***) :** 0.2361
- **RRMSE(**α***, 0.8** γ***) :** 0.2363

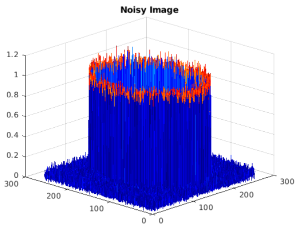
Discontinuity Adaptive MRF Prior

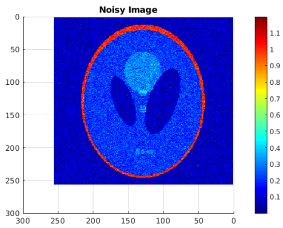
- α* **:** 0.993
- γ* **:** 0.0013
- **RRMSE**α***,**γ***) :** 0.2359
- **RRMSE(1.2** α***,** γ***) :** 3.1957e+29
- **RRMSE(0.8** α***,** γ***) :** 0.2740
- **RRMSE(**α***, 1.2** γ***) :** 0.236
- **RRMSE(**α***, 0.8** γ***) :** 0.236

Colormap Plots

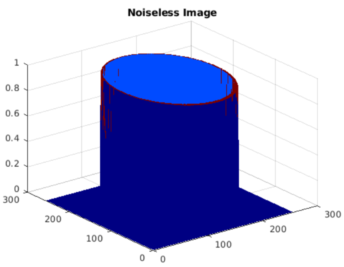
Side View and Top View respectively of the jet colormaps I obtained for the various images are as follows -

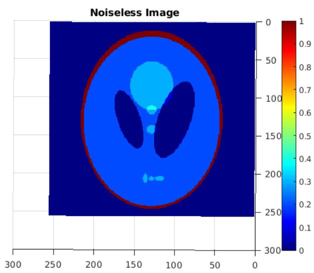
Noisy Image



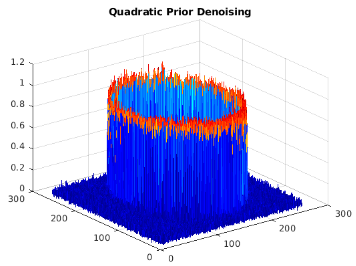


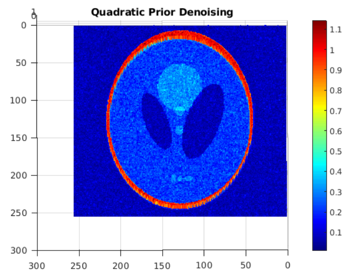
Noiseless Image



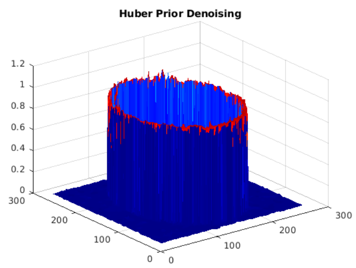


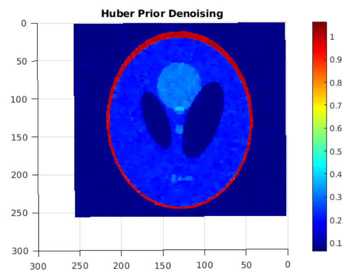
Quadratic Prior Image



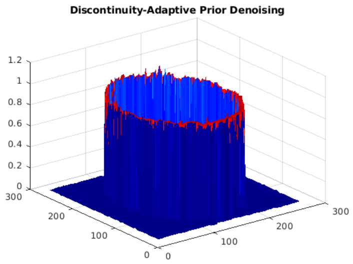


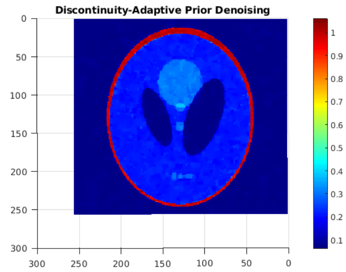
Huber Prior Image





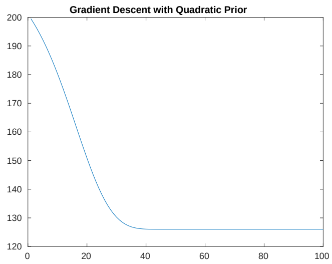
Discontinuity-Adaptive Prior Image

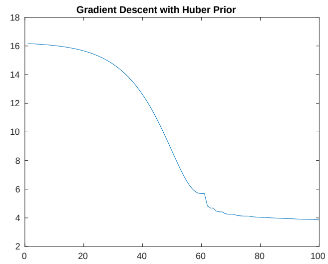


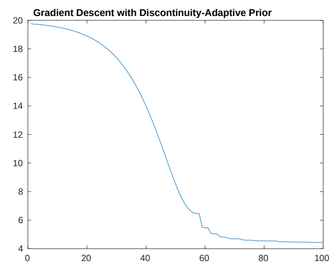


Objective Function Plots

The plots obtained for the objective function(sum of negative log posterior for all the pixels of the image) as we perform gradient descent are as follows -







Question 2

Noise Model : Gaussian

Results

RRMSE between Noisy and Noiseless image : 0.1424

Quadratic MRF Prior

- α* **:** 0.94
- **RRMSE(**α*) **:** 0.1270
- **RRMSE(1.2** α*) **:** 0.3761
- **RRMSE(0.8** α*) **:** 0.1284

Huber MRF Prior

- α* **:** 0.25
- γ* **:** 0.007
- **RRMSE**α*,γ*) **:** 0.1251
- **RRMSE(1.2** α*, γ*) **:** 0.1262
- **RRMSE(0.8** α*, γ*) **:** 0.1255
- **RRMSE(**α*, **1.2** γ*) **:** 0.1259
- **RRMSE(**α*, **0.8** γ*) **:** 0.1253

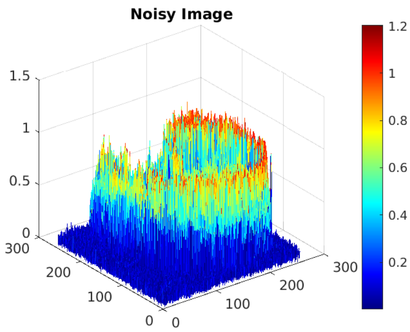
Discontinuity Adaptive MRF Prior

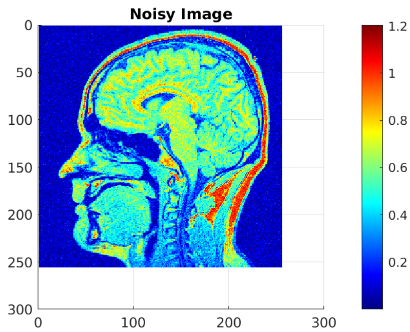
- α* **:** 0.48
- γ* **:** 0.005
- **RRMSE**α*,γ*) **:** 0.1155
- **RRMSE(1.2** α*, γ*) **:** 0.1177
- **RRMSE(0.8** α*, γ*) **:** 0.1173
- **RRMSE(**α*, **1.2** γ*) **:** 0.1159
- **RRMSE(**α*, **0.8** γ*) **:** 0.1156

Colormap Plots

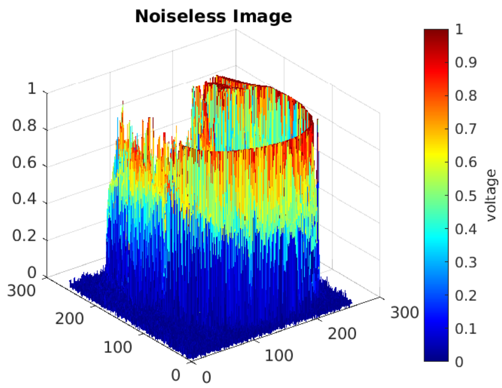
Side View and Top View respectively of the jet colormaps I obtained for the various images are as follows -

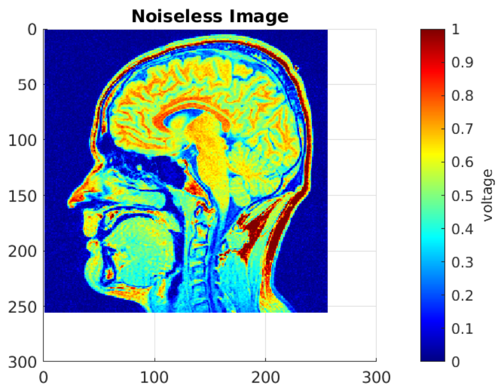
Noisy Image



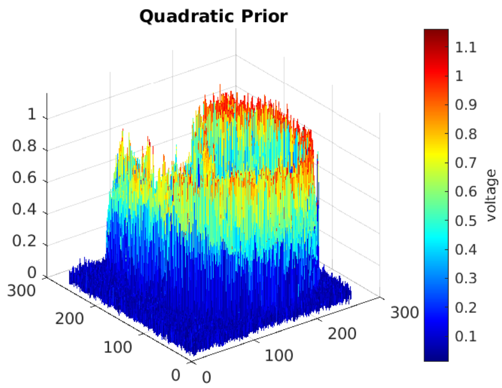


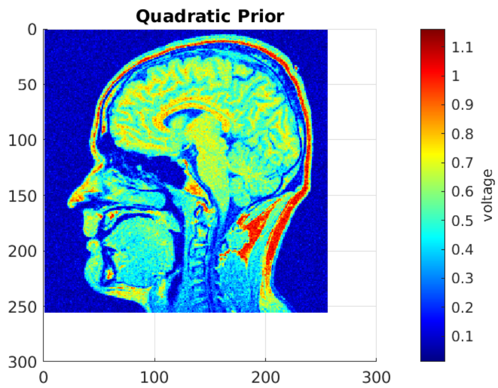
Noiseless Image



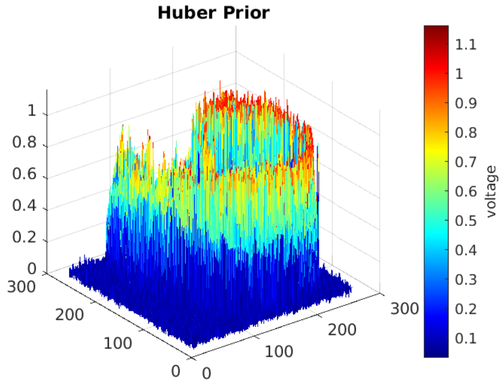


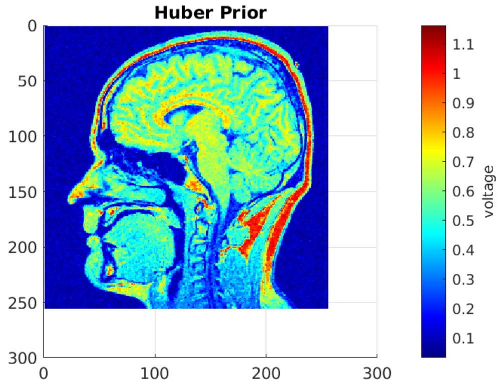
Quadratic Prior Image



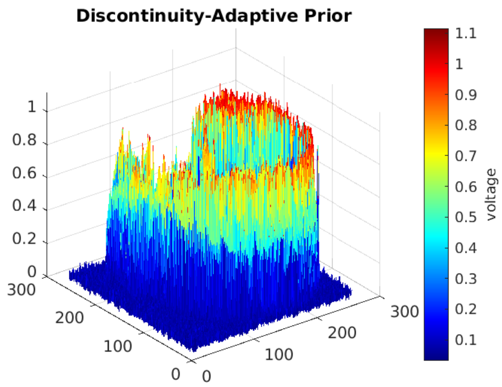


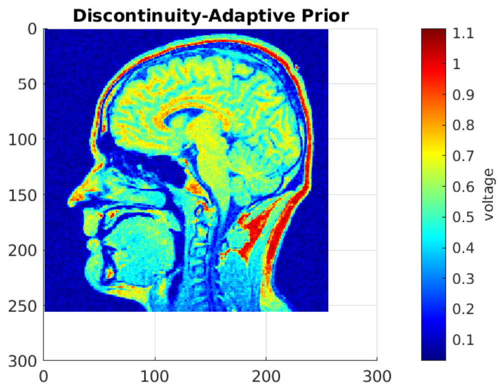
Huber Prior Image





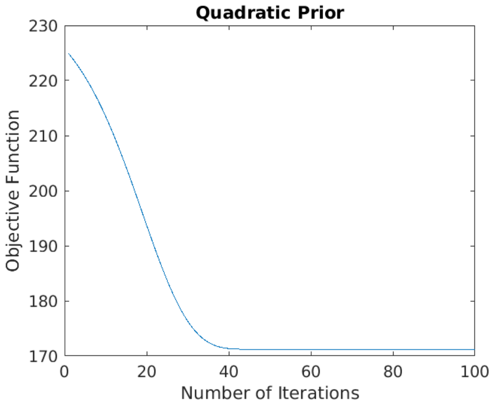
Discontinuity-Adaptive Prior Image

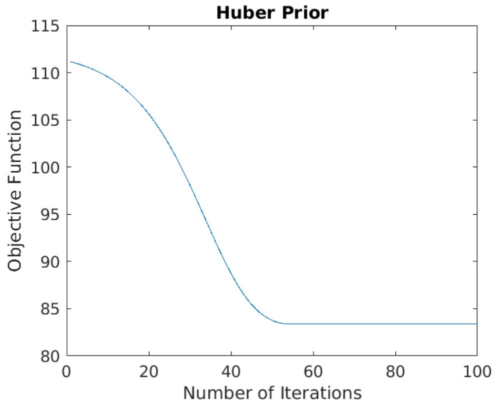


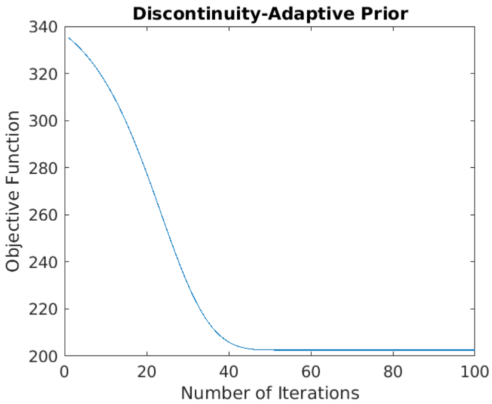


Objective Function Plots

The plots obtained for the objective function(sum of negative log posterior for all the pixels of the image) as we perform gradient descent are as follows -







Question 3

MRF Prior Model

Assuming that the intensity value associated with each pixel is a 3x1 vector as follows - $\begin{bmatrix} r \\ g \\ b \end{bmatrix}$ where r, g and b represent the color intensity of red, green and blue respectively. We can transform this vector into any other color space using a linear transformation applied by a 3x3 matrix A. This use a linear space helps us to account for the dependence between different components. The matrix A would be dependent on the application and could be learnt through experimentation. However, on a general note, we can consider A of the following form:

$$AX = \begin{bmatrix} r \\ g \\ b \end{bmatrix} - \frac{r+g+b}{3} = \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ -1/3 & 2/3 & -1/3 \\ -1/3 & -1/3 & 2/3 \end{bmatrix} X$$

Using this we obtain $t_i = Ax_i$. Further we know that as a MRF is a GRF we will have the probability distribution of the form -

$$P(T = t) = \frac{1}{Z} e^{\frac{-U(t)}{T}}$$

Here, Z is the sum of the exponential term over all pixels and $U(t) = \Sigma_{c \in C} V_c(t)$. Now in order to penalize large difference in t we can use the Huber penalty function where we put u^2 as sum of squares of the difference between the vectors in the same clique as the pixel we are considering.

$|u| = ||AX_i - AX_{N_i}||_2$

Note that penalizing u would penalize difference in angle between two pixel vector (i.e, difference in color) as well as magnitude of the two vector (i.e, change in pixel intensity viz regular penalty). Now we can use the Huber penalty function, to penalize any changes in neighbouring pixels after using a linear transformation and then using the magnitude of the vector we get. In this way we get the appropriate MRF prior model for Bayesian Denoising.

Noise Model

We can use a Multivariate Gaussian as the noise model for our Bayesian Desnoising Formulation. We represent the intensity at each pixel using a 3x1 vector as follows - $\begin{bmatrix} r \\ g \\ b \end{bmatrix}$ where r, g and b represent the color intensity of red, green and blue respectively. If we assume that the covariance matrix as C, we get the likelihood function as -

$P(Y_i|X_i) = G(Y_i|X_i, C) = \frac{1}{\sqrt{(2\pi)^3 det(C)}} e^{-\frac{(Y_i - X_i)^T C^{-1} (Y_i - X_i)}{2}}$

The covariance matrix can be obtained in a similar manner to how we obtained the value of variance in case of univariate Gaussian Noise, that is by using a prior scan of a phantom. This is the most general case, if we have prior knowledge that the noise in the three channels is independent the Covariance matrix will be a diagonal matrix with its elements equal to the independent variance along the three dimensions.

Bayesian Denoising Formulation

$$\begin{aligned} max_{X_i} P(X|Y,\theta) &= max_{x_i} P(Y_i|X_i,\theta) P(X_i|X_{N_i},\theta) \\ &= max_{x_i} \left(logP(Y_i|X_i,\theta) + logP(X_i|X_{N_i},\theta) \right) \\ &= max_{x_i} \left(\frac{-(AY_i - AX_i)^T C^{-1} (Y_i - X_i)}{2} + \sum_{c \in C} -V_c(T_c) \right) \\ &= min_{x_i} \left(\frac{(Y_i - X_i)^T C^{-1} (Y_i - X_i)}{2} + \sum_{c \in C} V_c(T_c) \right) \end{aligned} \tag{1}$$

$T_i := AX_i = A * \begin{bmatrix} r \\ g \\ b \end{bmatrix}$ A : Transformation matrix , T_i : New color space

- For gradient-based optimization with parallel updates, derivative at voxel i is:

  $\frac{\partial^{-log} P(X|Y,\theta)}{\partial X_i} = \Sigma^{-1}(X_i - Y_i) + \frac{\partial}{\partial X_i} \sum_{c \in C} V_c(T_c)$

- For entire image (n x n x 3 matrix) we can find the 3x1 gradient vector for each pixel
- Then we can update the pixels using stepsize τ at each iteration. Updated solution is:

  $X^{n+1} = X^n + \tau g(X)$