CS754 Project

190050037 Gaurav
190050053 Jayesh Singla

May 10 2021

1 DENOISING:

Figure 1: Original Image

First lets do with noisy image of $peak = 1$.

Figure 2: Noisy Image,peak=1 , $PSNR = 3.276$

Figure 3: Noisy Binned Image,kernel size =3,downsampling=2, $PSNR = 12.622$

The above image was first convolved with a kernel $ones(3,3)/9$ where the convolution operation maintained the original size of image. Then it was downsampled to half the dimension. This is done to increase the signal ration amidst the noise.

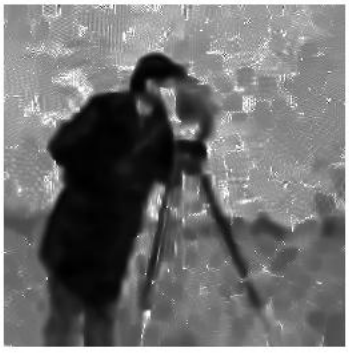


Figure 4: Denoised Image via BM3D, $PSNR = 17.953$

We improved from a PSNR value of from 3.276 to 17.953.

Figure 5: Denoised Image on the binned version, $PSNR = 20.41$

This shows that binning gives a better performance since the noised image which is getting feeded is quite improved thus improving the performance.

Figure 6: Denoised Image with P4IP, $PSNR = 20.55$

All these analysis was using noised added by using the function **poissonrnd**.

Hyperparameter tuning:

This tuning was applied to find the apt parameter β on a saturn image

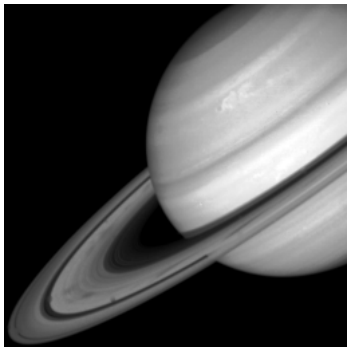


Figure 7: Noiseless image

Figure 8: Noisy image of peak =0.5

Now first a wide search for β was performed then a narrow search. Note that this was performed on a binned version.

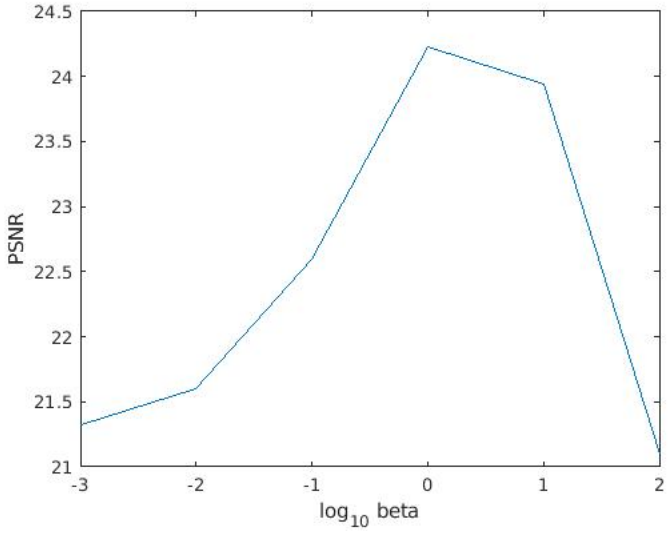


Figure 9: Wide search over β. Max at $\beta = 1$

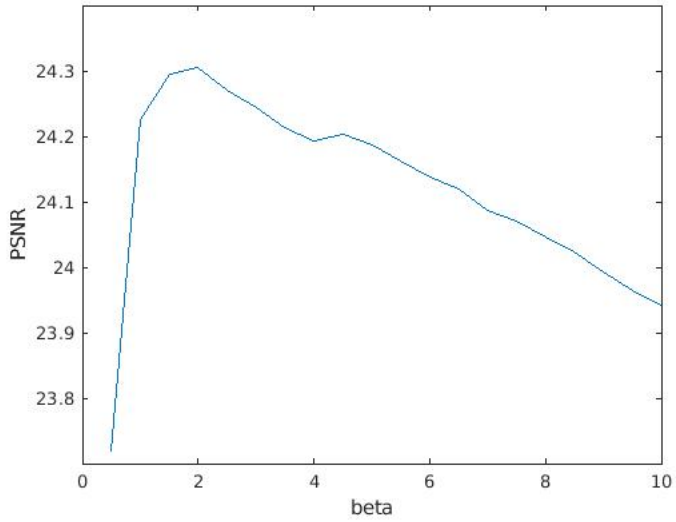


Figure 10: Narrow search over β. Max at $\beta = 2$

The Termination parameter(ϵ):

The parameter ϵ is used so that we need not have fixed number of iterations. This allows flexibility and makes it faster. This was the intention. But then it was observed that for not a good choice it makes things over smoothen or to say running for a fixed number of iterations can make image loose too much artifact on the pursuit of removing noise.

This analysis is also for the saturn image.

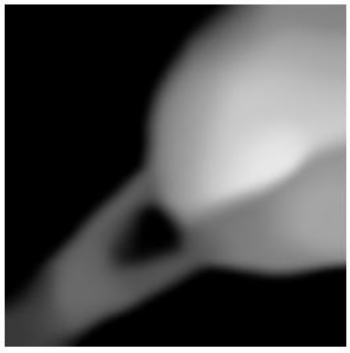


Figure 11: PSNR = 22.18. $\epsilon = 1e - 3$

If u observe its over smoothened so has lesser PSNR compared to

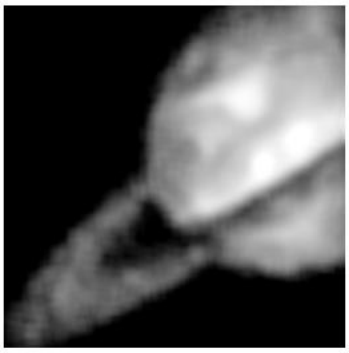


Figure 12: PSNR = 24.36 , $\beta = 2$, $\epsilon = 1e-2$

1.1 Tuning λ:

The performance is good for $\lambda = 0.25$. Although the tuning was performed . Tuning was performed for $beta = 2$ where $beta$ was fixed.

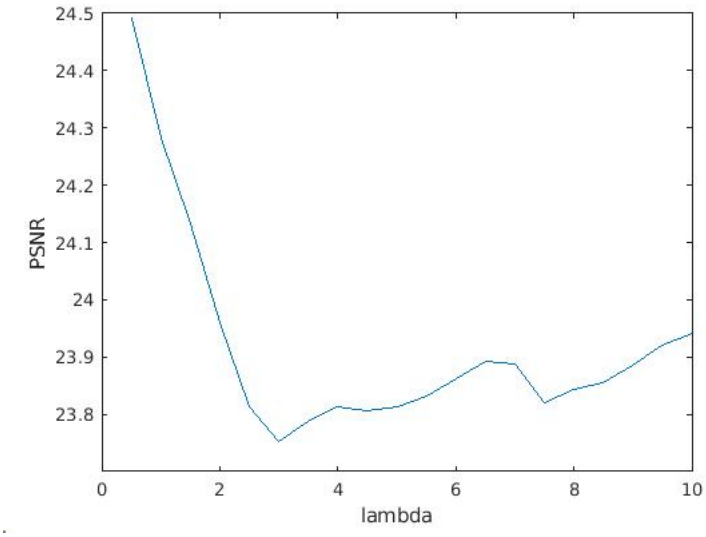


Figure 13: Wide search over λ. Max at $\lambda = 0.5$

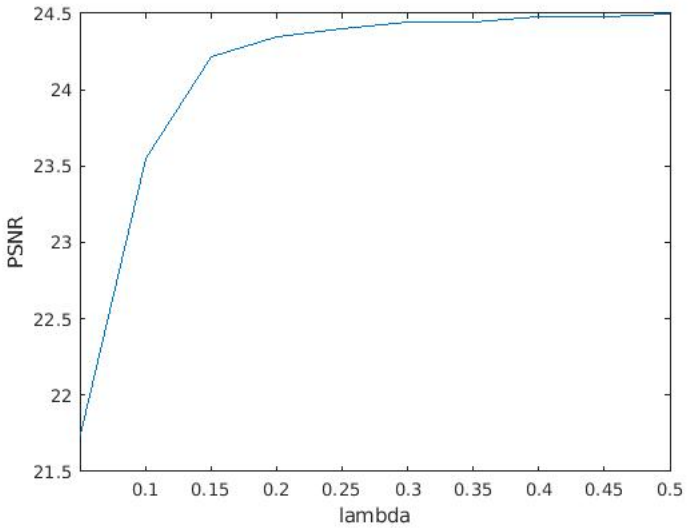


Figure 14: Finer search over λ.

Other Observations and Points:

- The performance was better for varying λ compared to fixed λ. The value that was used was 1.065 which was given in the paper
- The maximum number of iterations was set to 70 iter. Generally for $\epsilon = 0.01$ 70 iter wasnt needed and for lower ϵ or high β values not very high it was observed it reached 70. Although performance wise it never got better
- One important thing in binning was the conv kernel size and the downsampling size. For different images different values were needed the PSNR greatly differed by the value chosen. Sometimes it increased the PSNR by around 10-15 and sometimes it made it reduce. So this was a highly delicate and took quite a lot of time to get a parameter that gave a decent performance . Also binning was needed especially for low peak images

Brief Description of Code and its functions:

RRMSE:

We have defined two types of functions. Both have different implementations. Flag =1 is the modified version since at iteration number one the initialised image is zero. SO adjustment was made for tht. Flag =2 is the original RRMSE function.

Loading:

This function is to load the original image convert it to grayscale from rgb and also add poisson noise using **poissrnd** and return both the original image and the noisy image.

denoise:

This performs the first step in the ADMM iteration.

P4IP:

THis implements the algorithm. The ADMM has three steps in each iteration.

- First step is the one performed by denoise func which applies the correction for poisson based noise
- The second step is performed by BM3D package which is a gaussian denoiser

- Third step is just an update corresponding to the lagrangian

This algorithm breaks either when the no of iterations reach a limit which is 70 or the RRMSE btwn the current sol and prev sol is less than ϵ.

Main:

This just comprises of different implementations of the P4IP algorithm to try out different things.

This overall thing corresponds to denoising.

Experience and Scope of Improvement:

- It was quite good experience doing this project
- We had divided our work into two halves.One was denoising and the other was deblurring
- Initially there was a mistake which was due to inappropriate usage of imnoise function.
- The main code of algorithm was pretty easy to implement. Even the theory was also easy to understand
- The tuning and finding the right parameters to make the algo work was really challenging and took quite a long time since we are new to this.
- There was some time wastage due to lack of understanding how the pixel system of an image works and how the imshow functions and the PSNR functions work.
- Only BM3D was used. Can try other denoisers.
- SP3D wasnt implemented(Dictionary Learning Method). However it seems it runs so long and doesnt have a significant improvement in performance as told in the paper
- Better tuning of the parameters although we feel we have done pretty good job
- New innovations using this plug and play concept especially using one algorithm for another.

Deblurring using the P4IP

- Main difference : First ADMM step is not separable
- Will use an optimization method called L-BFGS to solve the optimization problem

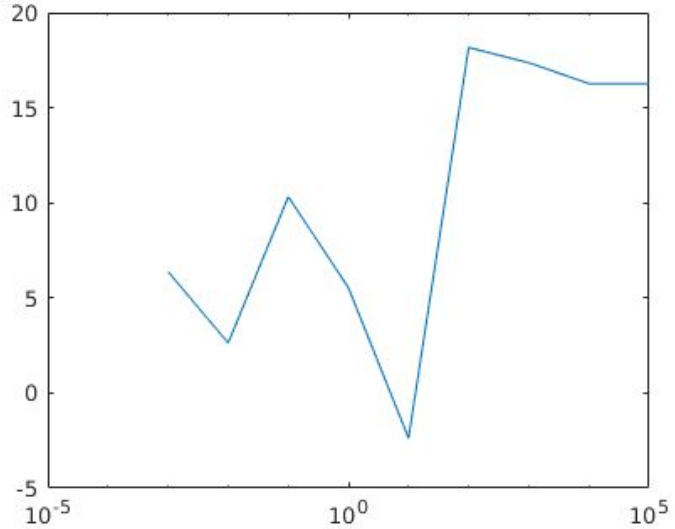
Hyperparameters

Similar to denoising, here we also have to tune hyper-parameters to get good results

We will tune **Lambda** and **Beta** both in two stages as before-

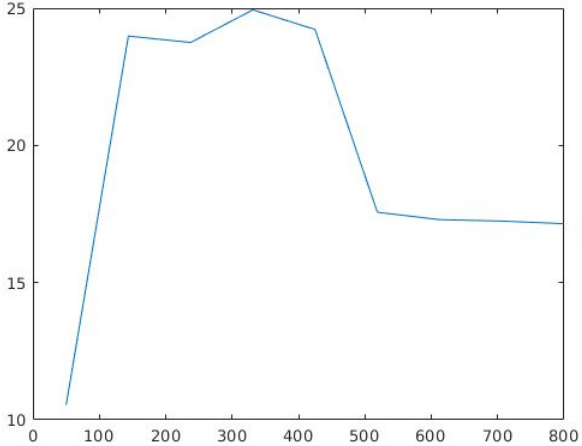
- Rough tuning
- Fine tuning

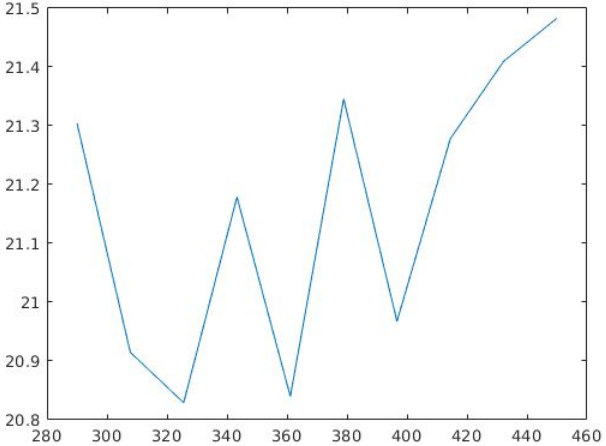
Lambda Tuning (Rough)

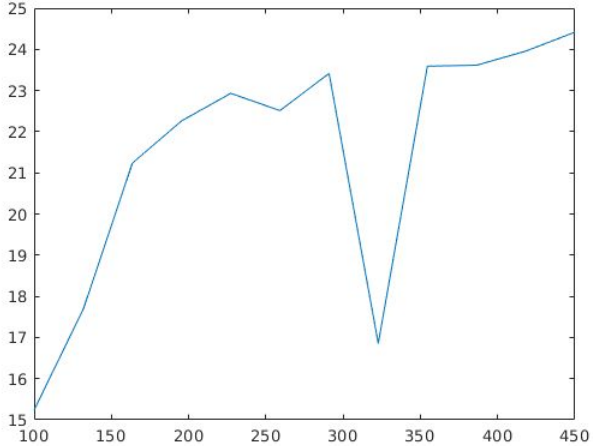


- Peak at lambda = 1

Lambda Tuning (Fine)

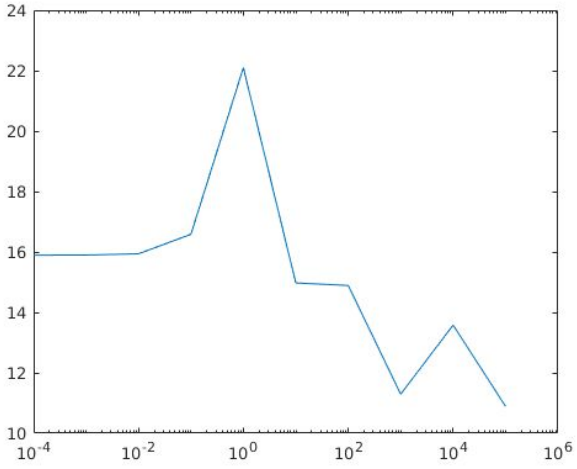




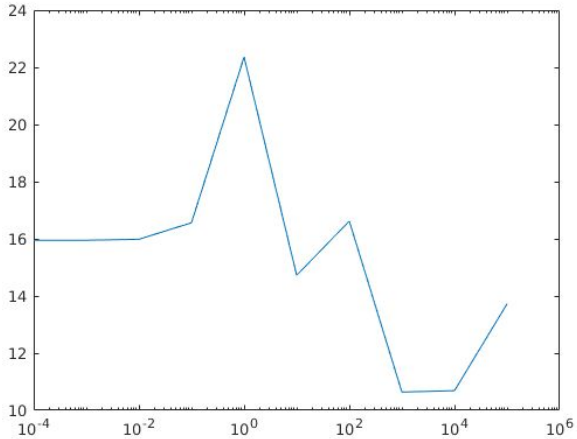


- Had to do fine tuning in three stages to get best PSNR
- Decreased epsilon as I fine tuned lambda
- Final lambda around 430-440 for peak noise = 1

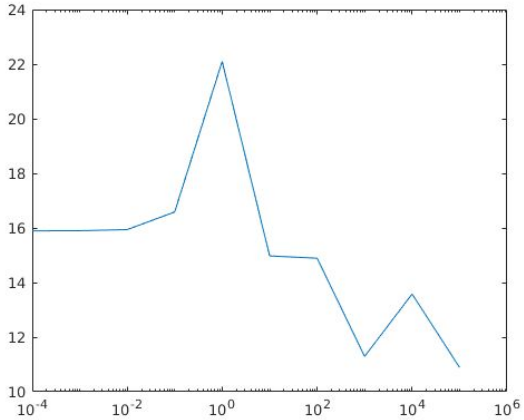
Beta Tuning (Rough)



ƛ = 350



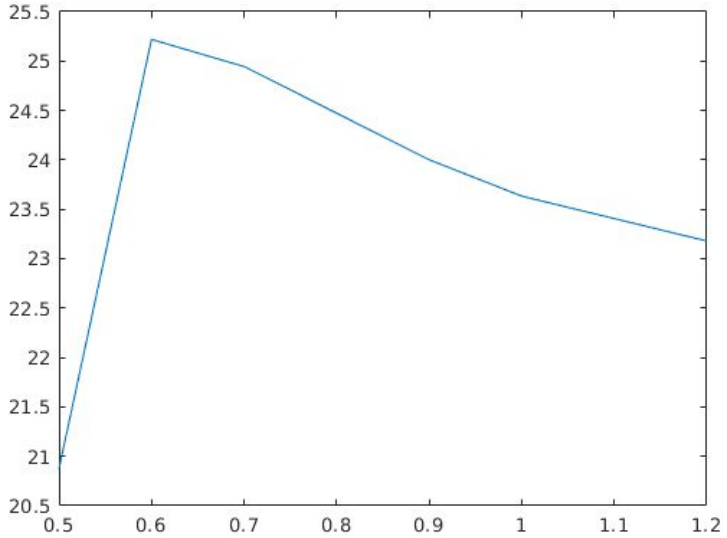
ƛ = 400



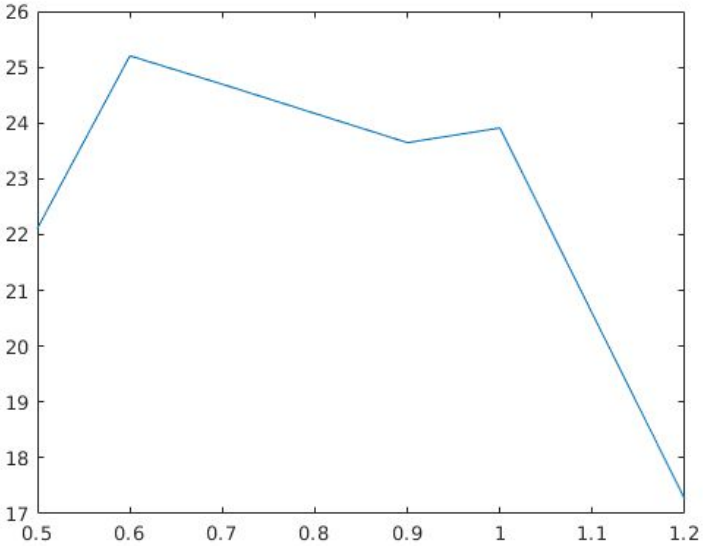
ƛ = 450

- Checked for multiple lambdas around optimal lambda to see trend
- Peak at around β = 1

Beta Tuning (Fine)



ƛ = 440



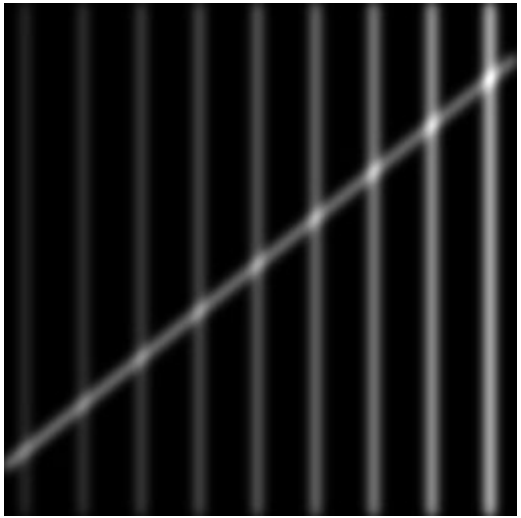
ƛ = 450

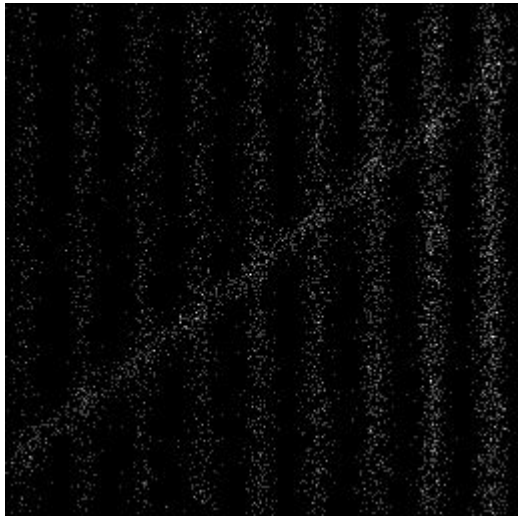
- Best lambda found to be 440-450
- Best beta found at 0.6

Results

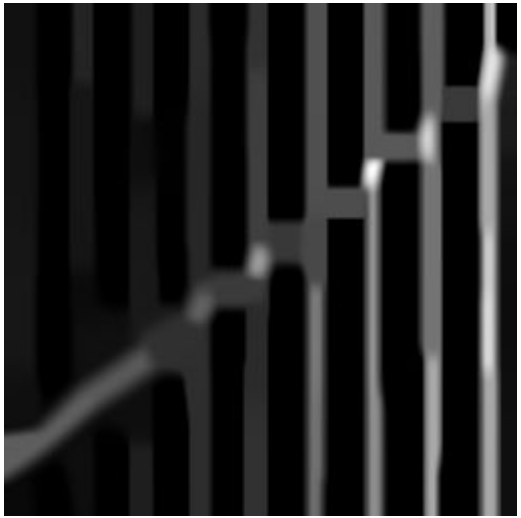
- Blur kernels used
  - 25 X 25 Gaussian kernel with $\sigma = 1.6$
  - 9 X 9 Uniform Kernel
  - 15 X 15 Custom kernel with entries generated as $1/(i^2 + j^2 + 1)$ for i,j = -7,-6...0..6,7
- We will show results on standard images like Peppers, Cameraman, Ridges etc

Custom blur with peak = 1



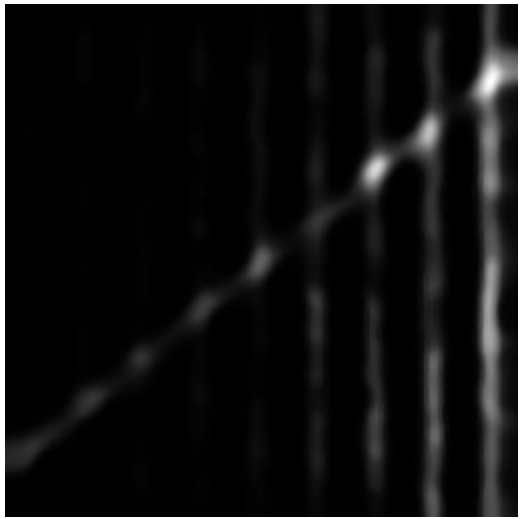


PSNR 16.1 dB



PSNR 22.62 dB

Custom blur with peak = 1 with BM3D only



PSNR 19.71 dB

This is worse than our reconstruction

Gaussian blur with peak = 2

PSNR 7.364 dB

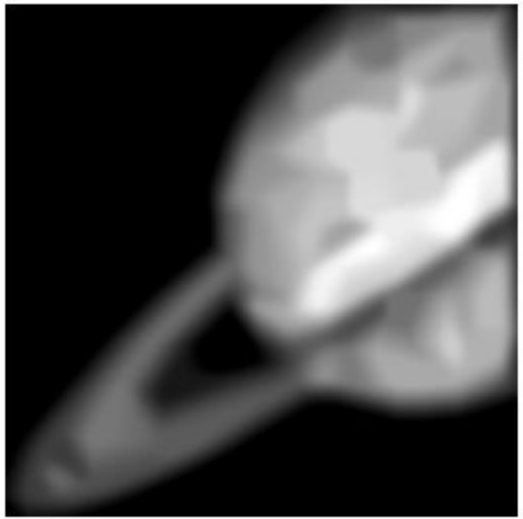
PSNR 19.59 dB

Uniform blur with peak = 1

PSNR 6.86 dB

PSNR 18.45 dB

Bad Initial Guess vs Good Initial Guess



PSNR 22.919 dB

Reconstruction at peak = 1

Kernel	Image	PSNR
Gaussian	Saturn	19.625283
	Pepper	17.967657
	Cameraman	17.929531
	Ridges	**26.439523**
	House	17.326899
	Curve	14.886446
Custom	Saturn	**20.50723**
	Pepper	**18.194646**
	Cameraman	16.039723
	Ridges	**19.649932**
	House	18.006444
	Curve	**19.273641**
Average	Saturn	**22.919237**
	Pepper	18.001448
	Cameraman	16.402292
	Ridges	**20.627435**
	House	17.850388
	Curve	19.067032

Observations/Inferences

- The hyperparameter values mentioned in the thesis of the authors do not work.It may be the case that their image representation(pixel value range) may be different which leads to different hyperparameters for us.
- As the ADMM step is not deterministic in this case, the reconstruction quality and number of iterations taken to converge is heavily dependent on the initial guess.

Improvements

- Get optimal beta, lambda values for different peaks and fit a curve to get a best general formula for the hyperparameters
- Try to derive an approximate closed form solution for the ADMM step to provide as an initial guess to L-BFGS