# Sparse Image Filtering with High Frequency Signal Preserved

Ruiqi Ma
Dartmouth College
Dartmouth College Hanover, NH
Ruiqi.Ma.GR@dartmouth.edu

**Figure 1.** This figure show the result of applying the filtering method proposed by this paper. The left image shows the input image. The right image shows the result image after filtering.

## Abstract

*Many different kinds of filter kernels can be applied to do image denoising. The main object of this paper is to denoise the low frequency background while preserving the edges and details of the high frequency components in a sparse image. An image is considered to be a sparse image if there are sparse high frequency components on a low frequency background, and the sparse high frequency signal is the part what to be preserved while denoising the low frequency background. Next step is to implement some of the approaches and find the one that can run in feasible time while producing desirable denoising result that preserves the high frequency signal in a sparse image. Fig. 1 shows the result of this paper using an example of star imaging,*

## 1. Introduction

Most denoising methods are based on the assumption that a low-pass filter is applied, and the high-frequency information is the part of the signal to get rid of. However, let's consider a case for star imaging. Each star usually only takes up 2 to 5 pixels in both width and height. Its intensity value is quite different from the background, which is the dark sky. Thus, stars can definitely be regarded as high-frequency signals that should be removed. However, these are actually the most important pieces of information we want to preserve in the images. The goal is to focus on denoising the dark background while preserving the light stars.

Several different filtering method are discussed in this paper, including Gaussian filtering, median filtering, bilateral filtering, joint bilateral filtering. Their run time efficiency theoretically. Sec. 2 discusses these different methods. In Sec. 3 and Sec. 4, two different approaches are proposed for the case of denoising star images. Sec. 3 introduces a potential method for denoising a sparse image and shows why it does not work. Sec. 4 introduces another approach that can successfully denoise the image while preserving the high frequency details in the sparse image.

## 2. Related work

One of the main applications for filtering is to denoise images. Some filters are edge-preserves, like bilateral filtering [9], while some others are not, like median filter [11] and Gaussian filter.

### 2.1. Gaussian kernel

The Gaussian kernel is named after Carl Friedrich Gaussian. It take two parameters, $x$ and $\sigma$, where $x$ is the input value and $\sigma$ controls the "range" that this Gaussian filter takes into consideration. The 1D version of Gaussian filter is:

$$G_{1D}(x,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{1}$$

The 2D version of Gaussian filter is:

$$G_{2D}(x,y,\sigma) = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

The larger $x$ is, the influence of the pixel that introduce $x$ has smaller influence on the final result of the kernel. The larger $\sigma$ is, the blurrier results Gaussian filter has. Gaussian filter works locally, and is not an edge-preserved filter, which means that the edge may also be blurred.

In our case, we want o preserve the details of stars, which means the edges should be finely restored and not blurry. Only applying Gaussian filter does not work in our cases.

### 2.2. Bilateral filtering

Bilateral filtering is a nonlinear filter introduced by Tomasi and Manduchi [9]. It has many applications such as denoising, texture and illumination separation [6], tone mapping [3], detail enhancement [5], data fusion [4], 3D fairing, etc.

The filter replace each pixel with a weighted average of its neighbors. The weights on the neighbors are calculated base on the multiplication of two Gaussian filters. Let $\mathcal{S}$ denote the spatial domain, and $\mathcal{R}$ denote the range domain. In plain English, $\mathcal{S}$ is the domain of the pixel location. For an image with width $w$ and height $h$, the spatial domain $\mathcal{S}$ is from 0 to $wh-1$. For a RGB image with intensity from 0 to 255, the range domain $\mathcal{R}$ is from 0 to 255. Bilateral filter takes both the difference between the pixel and its neighbors in both $\mathcal{S}$ and $\mathcal{R}$.

$$I_p^{bf} = \frac{1}{W_p^{bf}} \sum_{q\in S} G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(|I_p - I_q|) I_q \tag{3}$$

$$\text{with} \quad W_p^{bf} = \sum_{q\in S} G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(|I_p - I_q|) \tag{4}$$

The parameter $\sigma_s$ controls the spatial sensitivity of the filter. The larger $\sigma_s$ is, the larger size of spatial neighborhood is used to filter the pixel. The parameter $\sigma_r$ controls the pixel intensity sensitivity of the filter. The larger smaller $\sigma_r$ is, the less importance the pixels with larger intensity difference has.

**Separable bilateral kernel.** Separable bilateral kernel can be run in $O(|\mathcal{S}|r)$ time where $r$ is the kernel size. Instead of using a 2D kernel of size $r^2$, it apply a 1D kernel of size $r$ first vertically and then horizontally (the effect are the same if the order is changed). The main drawback of this approach is that it may cause fake vertical and horizontal lines. Especially in the sparse image case, the feature we want to preserve consists of smaller among of pixels, such fake effect may affect a lot in the final result.

**Signal processing grid.** The main concept of the signal processing grid method is to expend the representation of the image to higher dimensions. However, as mentioned by Paris et al [8], this approach is too slow if it's applied to RGB images and is using small kernel.

**Layered approximated bilateral filtering.** The main concept of this approach is do a pre-processing work that first we sample a set of intensities from the image, then we calculate the Gaussian filter on these intensities in range scale and store them in a Gaussian filter dictionary. While applying bilateral filtering to the pixel $p$, find the closest intensity $i_{closest}$ in the sampled intensity set, then use the Gaussian filter calculated for intensity $i_{closest}$ as the Gaussian filter for the intensity of pixel $p$. This approach is very fast, but as mentioned in this paper [2], the result from this approach can be very different from the result of running brute force approach. The inaccuracy and inconsistency are the main drawback of this approach and make it unsuitable for spare image filtering.

### 2.3. Joint bilateral filter

Joint bilateral filter is introduced by Eisemann and Durand [4] and Petschnigg et al [7]. It's a variant of bilateral filter that decouple the spatial Gaussian filter and the range Gaussian filter. It gets the spatial Gaussian filter $G_{\sigma_s}$ on an image $I$, and gets the range Gaussian filter $G_{\sigma_r}$ on another image $F$. One application is denoising using flash and no-flash images. The no-flash image $I$ has lots of noise and lack of details while the color intensity is near to what we want. The flash image $E$ has less noise and the edges are clearer, while it may contain some fake features such as shadows.

$$I_p^{jbf} = \frac{1}{W_p^{jbf}} \sum_{q\in S} G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(|E_p - E_q|) I_q \tag{5}$$

$$\text{with} \quad W_p^{jbf} = \sum_{q\in S} G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(|E_p - E_q|) \tag{6}$$

The intuition of joint bilateral filter is to make use of the property that edges are better detectable in the images of low noise. One prerequisite of applying joint bilateral filter on image $I$ and $E$ is that these two images need to be aligned first.

### 2.4. Median filter

The median filter was introduced by Tukey [11]. It simply assigns the median value in the kernel to the pixel. The

brute force approach of median filter is $O(|\mathcal{S}|r^2)$, where $r$ is the size of the kernel. Faster approach is introduced by Huang [10]. He used histogram to store the occurrence of pixel intensities, and update the histogram in $O(r)$ time for each pixel. Inspired by Huang, Ben Weiss [1] introduced an even faster algorithm to update the histogram in $O(\log r)$ time for each pixel.

## 3. Joint bilateral filtering with alignment

In this section, I'm going to discuss several potential approaches to denoise sparse images. We consider an image to be a sparse image if there are sparse high frequency components on a low frequency background, and we want to preserve the sparse high frequency signal. Through out this section, the case where we want to denoise a star image is introduced. The main object is to mainly denoise the low frequency background while preserving the edges and details of the high frequency components.

We take two images $I$ and $E$. Image $I$ should be of larger ISO, which we say that it corresponds to the no-flash image in the previous example. Image $E$ should be of longer exposure time but with lower ISO, corresponding to the flash image in the previous example.

### 3.1. Alignment

The prerequisite of applying joint bilateral filtering is that two images should be aligned. An assumption is that images $I$ and $E$ are taken from the same position and angle during a short time interval. You probably want to take two images sequentially with different settings by the same camera and with its position and orientation unchanged. We assume all other stars are rotated around the north star, and the translation can be ignored. Thus, we only need to find the rotation matrix before we have two images aligned.

Suppose we include the north star in our images. One approach is to first find the position of north star using the fact that the position of the north star remains unchanged, we can regard it as the center of rotation, then manually choose one pixels in image $I$ and $E$ respectively. These two manually-chosen pixels ideally should be the corresponding points of the same star. With these two points and the position of north star, we should be able to calculate the rotation matrix. Another approach is to use manually-chosen pixels only, considering the situation where the north star is not included in the image. Using sufficient among of manually-chosen pixels, we should also be able to calculate the rotation matrix. The third approach is to use the alignment packages to run the feature-based alignment algorithm.

However, alignment is hard to do in real star images for the following reason. In order to get accurate rotation matrix, we expect the pixels that are manually chosen should be the corresponding pixel pair in ground truth. However, it's not feasible to expect manually-chosen points to be exactly the ground truth pair. A slight issue of one pixel while choosing may lead to bad result, since most of the stars shown on the image only take up 2 to 5 pixels. One pixel difference is huge for such small component while applying $G_{\sigma_r}$ to it. Moreover, due to the "shining" fact of stars, the shape of stars itself may differ in different images. In other images that has some more detectable features and want to apply the sparse image filter to some part of the image, you may consider to use the feature based alignment algorithm first and apply the joint bilateral filter to it. In the star image case, there's not much features and it's a low contrast image, applying feature based alignment algorithm doesn't work.

### 3.2. Joint bilateral filtering

With bad alignment, joint bilateral filter can not be well applied.

## 4. Bilateral filtering with mask

### 4.1. Notations

This section explains the notations used in the rest of the paper. $\mathcal{S}$ denotes the set of pixel locations, it is the spatial domain of an image. $|\cdot|$ denotes the size of a set. $\|\cdot\|$ the denotes l2 norm of a distance. For instance, for an image $I$ of height $h$ and width $w$. $|\mathcal{S}| = wh$. $\mathcal{R}$ denotes the set of all possible pixel intensities, it is the range domain of an image. $G_{\sigma_s}$ denotes the Gaussian filter in the spatial domain with parameter $\sigma_s$. $G_{\sigma_r}(x)$ denotes the Gaussian filter in the range domain with parameter $\sigma_r$. $I_p$ denotes the pixel intensity for pixel $p$ in image $I$. $I^{bf}$ denotes the output image after applying the filter to image $I$.

### 4.2. Pre-processing

To avoid the alignment problem, we consider to use only one image. The goal for doing pre-processing is to get a mask for the star pixels. A pixel is considered as a star pixel if it's regarded as a pixel in the star component. A mask for star pixels is needed, for the reason that bilateral filtering should not be applied to these pixels. Figure x shows a comparison of the denoise results of the same star image with the mask applied and not applied.

The method proposed for finding the region of the mask is based on the pixel intensities. For the rest of section, unless it's specially mentioned, all the operations are on a gray scale image. The brightest pixels in a star image is always considered as star pixels. As the brightness decreases, a pixel is more unlikely to be a star pixel. A threshold $threshold$ is introduced. The pixels with intensity $> threshold$ is considered to be a star pixel with probability 1. Moreover, a threshold $extensionThreshold$ is introduced to function as the lowest intensity a star pixel

can have. A pixel is considered to be a potential star pixel if its intensity is $> extensionThreshold$. However, only the potential star pixels that is the neighborhood of a star pixel can be considered to be a star pixel. Algorithm 1 shows the pre-processing process.

**Algorithm 1** Pre-processing

$\mathcal{S}_{star} \leftarrow empty$
**for** each pixel $p$ in $I_{gray}$ **do**
  **if** $I_{gray}[p] > threshold$ **then**
    add $p$ to $\mathcal{S}_{star}$
  **end if**
**end for**
$\mathcal{S}_{extension} \leftarrow empty$
**for** each pixel $q$ in $\mathcal{S}_{star}$ **do**
  **for** each pixel $m$ in $q$'s neighbors **do**
    **if** $I_{gray}[m] > extensionThreshold$ **then**
      add $m$ to $\mathcal{S}_{extension}$
    **end if**
  **end for**
**end for**
$\mathcal{S}_{star} \leftarrow$ union of $\mathcal{S}_{star}$ and $\mathcal{S}_{extension}$
**return** $\mathcal{S}_{star}$

### 4.3. Bilateral Filtering

The basic approach of bilateral filtering is to apply a kernel on each pixel of the image in tiles. To better understand the process of bilateral filtering and to apply the mask to it, a version of bilateral filter is implemented on my own. The Gaussian filter with parameter $\sigma$ on input value $x$ is:

$$g(x) = \exp(-\frac{x^2}{2\sigma^2})/(2\pi\sigma^2) \tag{7}$$

The Gaussian filter in spatial domain is constant for each pixel as the kernel size are the same for all the pixels. Note that to avoid unexpected color change, it's necessary to do bilateral filtering on the luminance channel. After filtering, it should be converted back to RGB space. Since the noise pixels can be either lighter or darker than the ground truth value of that pixel, after applying the filter to the background part which has no mask, the background can be lighter than expected. A parameter $darkConstant$ is introduced to modify the luminance channel of the background. The equation for getting the filtered value is:

$$I_p^{bf} = \begin{cases} I_p & p \in S_{star} \\ \sum_{q \in k(p)} G_{bf} I_q / (darkConstant \cdot W_p) & o/w \end{cases} \tag{8}$$

where $G_{bf} = G_{\sigma_s} G_{\sigma_r}(|I_p - I_q|)$ and $k(p)$ denotes the pixels set of the kernel centered at $p$.

Algorithm 2 shows the process for applying bilateral filter. Figure x shows the comparison of the denoising result after applying $cv2.bilateralFiltering$ and the self-implemented version.

**Algorithm 2** bilateral filter with mask

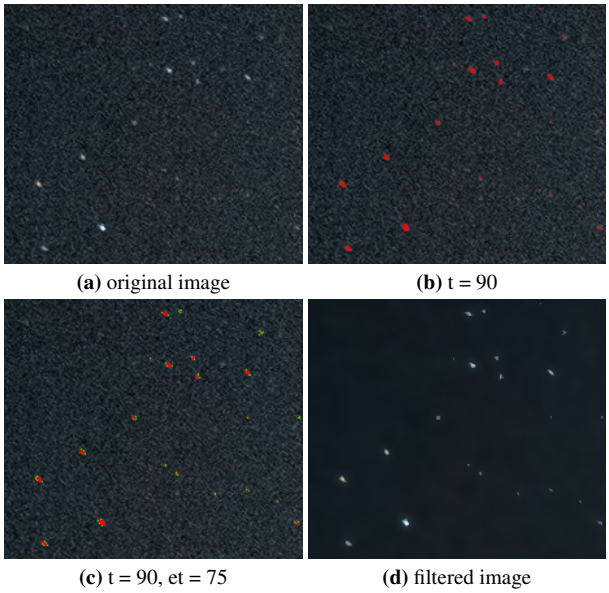
$I^{bf} \leftarrow 0$
$points \leftarrow$ meshgrid of range$[-1, 1]$ and range$[-1, 1]$
$l2Norm \leftarrow \|points - [0, 0]\|$
$G_{\sigma_s} \leftarrow gaussianFilter(l2Norm, \sigma_s)$
**for** each pixel $p$ in $\mathcal{S}$ **do**
  **if** $p \in S_{star}$ **then**
    $I_p^{bf} \leftarrow I_p$
  **else**
    $W_p \leftarrow 0$
    **for** each pixel $q$ within the kernel centered at $p$ **do**
      $w \leftarrow G_{\sigma_s} G_{\sigma_r}(|I_p - I_q|)$
      $I_p^{bf} \leftarrow I_p^{bf} + w I_q$
      $W_p \leftarrow W_p + w$
    **end for**
    $I_p^{bf} \leftarrow I_p^{bf} / (W_p \times darkConstant)$
  **end if**
**end for**

## 5. Results

In this section, denoising results for using different parameters and approaches are shown and compared. Fig. 2 shows the results of the star pixels after applying different $threshold$ and $extensionThreshold$. Comparing the input image and the star pixels found, the accuracy of the algorithm is feasible.

Fig. 3 compares the result after applying traditional bilateral filter using $cv2.bilateralFilter$ function from $cv2$ package and the result after applying the bilateral filter with mask approach proposed by this paper. As we can see through the comparison, the bilateral filter with mask can preserve star signals in star imaging. However, the time complexity of the self-implemented bilateral filtering is much higher than that of $cv2.bilateralFilter$. The latter takes 1 second, while the former one may take around 30 minutes. Time complexity is a trade-off for the good result. Also, the parameter tuning is the key part for this star image denoising. Since many different parameters can be using for different star imaging, two thresholds vary in different images. Appropriate values for these parameters need to be found to have a good filtering result.

This approach may fail when very high ISO is used or very low ISO is used to capture the image. The former will cause the image to be too noise that the signal-to-noise ratio may be low. Noise may suppress the brightness of the star so that the algorithm can not accurately tell where the stars are. The latter will cause the star in the image be too dark, even some signal may be lost and not well captured. In this

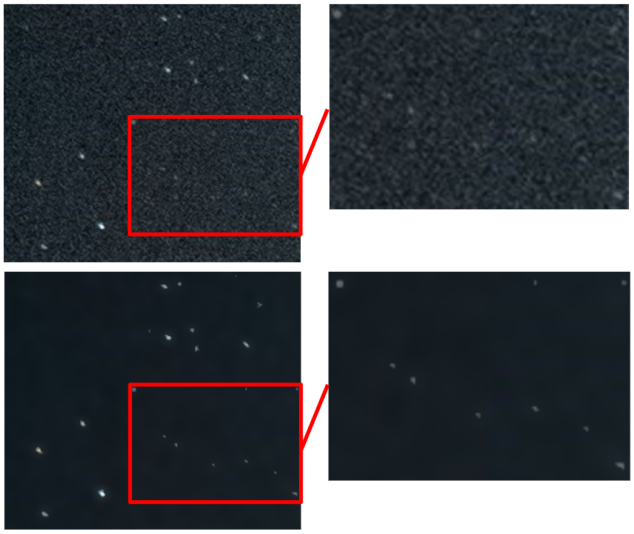


**(a)** original image **(b)** t = 90

**(c)** t = 90, et = 75 **(d)** filtered image

**Figure 2.** This figure shows the inter-results and the final results after applying the bilateral filtering with mask. $t$ represents the $threshold$ parameter, $et$ represents the $extensionThreshold$ parameter. The parameters are: $t = 90$, $et = 75$, the kernel size for the bilateral filter is 15, $\sigma_s = 4$, $\sigma_r = 0.1$, $darkConstant = 1.5$. Fig. 2a shows a small portion of the original input image. Fig. 2b shows the result for the star pixels. The red pixels represents the star pixels. Fig. 2c shows the result after doing the star pixel extension. The green pixels represents the star pixels added during the extension. Fig. 2d show the final result after applying filtering with mask and darkness adjustment using the parameters above.

case, the algorithm may not be able to tell there's a star or not since the image does not give it such information.



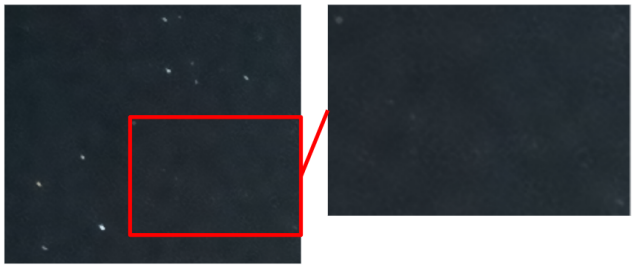


**Figure 3.** This figure shows how my approach can preserve the star details while having good denoise effect. The first figure shows the original input image. The second figure shows the result after applying the bilateral filtering with mask. Fine details of stars are preserved and the background is smooth, The third figure show the result after applying $cv2.bilateralFilter$ to the input image. Star details are lost here.

## References


[1] Ben Weiss Shell; Slate Software Corp., Ben Weiss, Shel; Slate Software Corp., Shell; Slate Software Corp.View Profile, and Other MetricsView Article Metrics. Fast median and bilateral filtering, Jul 2006. 3

[2] Durand, Frédo, and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, 2002. 2

[3] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 21(3):257–266, 2002. 2

[4] E. Eisemann and F. Durand. "flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 23(3):673–678, 2004. 2

[5] R. Fattal M., Agrawala, and S. Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics*, 26(3):51, 2007. 2

[6] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. *Proceedings of the ACM SIGGRAPH Conference*, 12(1):433–442, 2001. 2

[7] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, , and K. Toyama. Digital photography with flash and no-flash image pairs. *Proceedings of the ACM SIGGACM Transactions on Graphics*, 23(3):664–672, 2004. 2

[8] Paris Sylvain and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision*, 81(1):24–52, 2007. 2

[9] Tomasi, C. Manduchi, and R. Bilateral filtering for gray and color images. *IEEE TPAMI*, pages 839–846, 1998. 1, 2

[10] HUANG T.S. Two-dimensional signal processing ii: Transforms and median filters. *Berlin: Springer-Verlag*, pages 209–211, 1981. 3

[11] J.W TUKEY. Exploratory data analysis. 1997. AddisonWesley. 1, 2