Sparse Image Filtering with High Frequency Signal Preserved

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

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. 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 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 accurate denoising result.

1. Introduction

Most denoising method is based on the assumption that we want a low-pass filter that high frequency information is the signal we want to get rid of. However, consider a case where we're doing star imaging. Each star usually only take up 2 to 5 pixels in both width and height, and its intensity value is quite different from the background which is the dark sky. Thus, stars can be definitely regarded as high frequency signal that will be removed. However, these are actually the most important information we want to preserve in the images. We want to focus on denoising the dark background while preserving the light stars.

I observed several different filtering method, including Gaussian filtering, median filtering, bilateral filtering, joint bilateral filtering. I compared their run time efficiency theoretically, and implement the brute force bilateral filtering. I also introduce a method that combines median filter and bilateral filter together to accelerate the algorithm. Next step I am going to see the actual run time difference and compare their denoising results among different filtering methods.

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 σ, where x is the input value and σ 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 σ 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 σ_s controls the spatial sensitivity of the filter. The larger σ_s is, the larger size of spatial neighborhood is used to filter the pixel. The parameter σ_r controls the pixel intensity sensitivity of the filter. The larger smaller σ_r is, the less importance the pixels with larger intensity difference has.

This paper (is going to) test on several different implementations of bilateral filter to compare the run time and the denoising effect of different approaches. I will discuss these approaches in the next section in detail.

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_{σ_s} on an image I, and gets the range Gaussian filter G_{σ_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. Method

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.

3.1. Joint bilateral filtering with alignment

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.

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_{σ_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.

Joint bilateral filtering. With bad alignment, joint bilateral filter can not be well applied. This section may be expended if the alignment problem is solved in the future.

3.2. Bilateral filter

To avoid the alignment problem, we consider to use only one image.

Brute force. The brute force approach is to apply a kernel on each pixel of the image in tiles. The feature we care about is star details, thus we need to choose small kernel size and small σ_s. The pixels that we regard them as stars should have large intensity difference to its surrounding neighbors. Thus σ_r should not too large and not too small since we also want to preserve the "blurred" edge of stars.

Algorithm 1 Brute force bilateral filter

```
$I^{bf} \leftarrow 0$
for each pixel p in $\mathcal{S}$ do
    $W_p \leftarrow 0$
    for each pixel $q$ within the kernel centered at $p$ do
        $w \leftarrow G_{\sigma_s}(\|p - q\|)G_{\sigma_r}(|I_p - I_q|)$
        $I_p^{bf} \leftarrow wI_q$
        $W_p \leftarrow W_p + w$
    end for
    $I_p^{bf} \leftarrow I_p^{bf}/W_p$
end for
```

The brute force algorithm works, but the run time is poor. Running on an approximate shape 250×350 RGB image with kernel size 7, it took around 70 seconds. In real application, the image size are often at least 10 times larger in both width and height, which may result in a run time more than 100 times larger. We need to find a faster implementation of bilateral filtering to make it more efficient.

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. Images and results after applying separable bilateral kernel may be added later.

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. Refer to future work section for more details.

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.

3.3. A combination of median and bilateral filter

The main down side of bilateral filter is its computational complexity. The median filter can be run in $O(r)$ time for each pixel, but it can not preserve edges. Thus, we consider a combination of median and bilateral filter. That is, for the most part of the image, which is the low frequency background in sparse image, median filter is applied. For the part that includes the edges of stars, bilateral filter is applied. In short, while moving the kernel to the next pixel, we decide whether the newly included pixels have very large difference in their intensity values. If so, we say that there may be an edge here, bilateral filter should be applied. Otherwise, we consider the kernel to be fully on the background, median filter is applied. The choice of choosing the threshold that we use to decide whether there's an edge or not is the key park of this algorithm. This is the main part of the highest priority in the future work.

4. Future work

Currently, mainly the theoretical parts are done. The brute force approach of the bilateral filtering and parts of the alignment is done for coding. Later, some implementation for the bilateral filtering approaches mention in section 3 will be done. Section 3.3 is of the higheest priority. Then the run time and the output quality will be compared. More images will be added to better explain the process and the results will be added in the future.

References

[1] Ben Weiss Shell; Slate Software Corp., Ben Weiss, Shel; Slate Software Corp., Shell; Slate Software Corp.View Pro-

file, and Other MetricsView Article Metrics. Fast median and bilateral filtering, Jul 2006. 2

[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. 3

[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. 1

[4] E. Eisemann and F. Durand. “flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 23(3):673–678, 2004. 1, 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. 1

[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. 1

[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. 3

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

[10] HUANG T.S. Two-dimensional signal processing ii: Transforms and median filters. *Berlin: Springer-Verlag*, pages 209–211, 1981. 2

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