Josiah Vincent

BILATERAL FILTER

COMPUTER PHOTOGRAPHY FINAL REPORT

ABSTRACT

The bilateral filter seeks to provide a method of image smoothing while still retaining the edges. It does this by replacing the intensity of each pixel with a weighted average of intensity values of nearby pixels. The weights are calculated using a Gaussian distribution model. The Gaussian calculates the weights using both the Euclidean distance between the pixels and the difference in the radiometric intensities.

INTRODUCTION

Filtering an image is an essential tool to have in your belt when working with an old or noisy image. There are many different kinds of filters that can be used for a wide variety of applications.

One of the most popular filters is the Gaussian smoothing filter. This filter is used to reduce image noise and essentially blends a pixel with its neighbors.

The reduction in image noise and the smoothing in general can be desirable, but the issue with gaussian blur is well, the blur. There are times when we want to retain the edges of an image. Especially when we are applying a filter of the whole image and not just parts of the image, we need to be able to make out the main shapes and features of the image while still blending and removing the undesired noise.

There are other filters such as a low-pass filter which essentially just takes the average of neighboring pixels which in turn smooths the image by decreasing the disparity between the pixel ranges.

The bilateral filter borrows from both and tries to take the best of both worlds by smoothing using a weighted average of nearby neighboring pixels while at the same time preserving the edges of the pixels.

This produces a much more desirable final result where you are able to remove some if not all of the image noise while still being able to easily make out the main features of the image.

METHOD

As it turns out the implementation of the bilateral filter is not too difficult. As declared above it is similar to implementing a combination of a low-pass filter and a Gaussian blur filter.

The general process of a bilateral filter is to go pixel by pixel over the image and applying the Bilateral Filter on each image.

Josiah Vincent

Because we need to use a mask of the form nxn this means that our final output image is going to have a margin on all sides that is half the diameter of the mask.

This means you are going to be starting at pixel (ceil(n/2), ceil(n/2)). You also need to make sure that you stop to leave room for the mask on the right side and also as you get to the bottom.

Once you have established the mask size you need to take into consideration a few other values that will be provided before the program start. Each of these are variances.

The first is a spatial variance. Spatial variance controls the smoothing over the spatial distance between 2 points.

The other is range variance. Range variance controls the smoothing over intensity differences.

As I said both of these are provided by the user and remain constant throughout the process of application.

Once you have the variances and the mask size that takes care of all the unknowns and we can get into the equations used to apply the filter.

The first equation here is the meat and potatoes of the whole thing.

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

Here I^filtered(x) is just the filtered pixel so this will need to occur for each . To break it up we can just ignore the stuff before the sum. Next checking out the bounds of the sum it is xi that exists in capital sigma. Each xi is just a pixel within the mask. The mask is denoted by capital sigma. So it is just saying take the sum of each pixel in the mask using the following.

After the sum I(xi) is just the intensity at the current pixel in the mask

f_r(||I(xi)-I(x)||) is the Gaussian distribution shown below where the input for x will be the Euclidean distance between the intensities of the central pixel in the mask and current pixel we are comparing against in the mask. The sigma value in the Gaussian distribution will be the range variance.

g_s(||xi-x||) is the Gaussian distribution show below where the input for x will be the physical Euclidean distances between the central pixel and the mask and current pixel we are comparing against in the mask. The sigma value in the Gaussian distribution will be the spatial variance.

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

We will be dividing everything discussed above by W_p. The only difference between this and the above is that we are not multiplying by I(xi). I will not explain everything again.

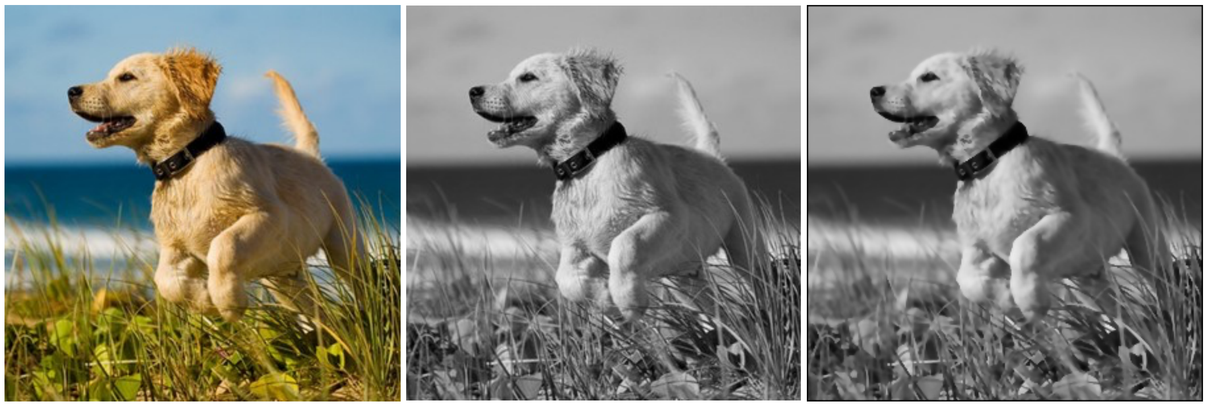
$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The final equation is the Gaussian distribution function that we were referencing before. Here the sigma value is going to be either the spatial or range variance value. The (x-mu) on the right side will just be replaced with the Euclidean distance of either the intensity or physical distance.

The I^filtered(x) will be applied over every pixel in the image to produce a new image.

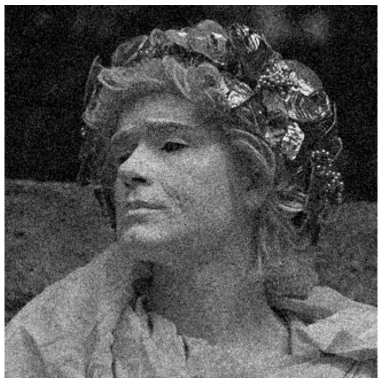
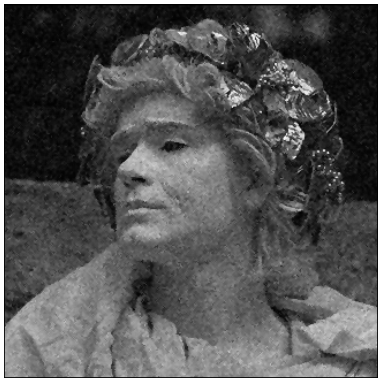
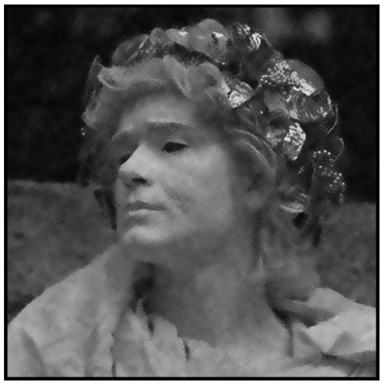
EXPERIMENTS

When doing experiments, it is useful to know that it is not always intuitive when tuning the variance values. It may take many attempts to get the image tuned properly.



The first example is good to get the general idea of what is going on. This example uses a 3x3 mask, a spatial variance of 50 and range variance of 25.

The original is a 300x300 image. We convert it to grayscale using opencv. Then we apply the filter over every pixel in the image. You can see how a lot of the sharp detail in the dog's fur is removed, but the edges are preserved so the main features of the image are still easy to see.

The second experiment I did was with noise. Here I picked an extremely grainy/noisy image just to see what the filter could handle. The results were ok. While I was definitely able to remove some noise in order to remove an acceptable amount of noise, I had to increase the mask size and variances which cause a loss of detail. I concluded the algorithm is probably better to handle small amounts of noise.

The image on the left is the original image the middle is using a 3x3 mask, spatial variance of 50 and range variance of 25. The right image is using a mask size of 7, spatial variance of 25 and range variance of 40.

LIMITATIONS

As discussed throughout the paper you may have noticed a few limitations with this method. The first limitation is time. Because we are having to go pixel by pixel it essentially comes down to being nested loops and this is not very quick. We are able to speed this up by utilizing the GPU, but that is something that was not implemented in this paper.

Another limitation is the detail of the image. The more you are having to smooth out and correct the more detail is going to be lost. While I was able to remove a substantial amount of noise you sometimes lose the sharp edges and there is really nothing you can do to tune this with this method.

Finally, the borders. If you are using a large mask on a low-res image you are going to have a black border that is half the diameter of your mask. This is not a big deal for high-res images, but for low-res it is something to consider.

CONCLUSION

The bilateral filter is a relatively easy method to implement that is good at doing lighter smoothing while still maintaining good edges to retain the features of the image. The method is good at smoothing out images that are too sharp or images with some noise.