Storage Efficient NL-Means Burst Denoising for Programmable Cameras

Brendan Duncan
Stanford University
brendand@stanford.edu

Miroslav Kukla
Stanford University
mkukla@stanford.edu

Abstract

An effective way to reduce noise in images involves taking a burst of snapshots, aligning them, and averaging them together. However, a burst of photos takes up a lot of memory, and most users only take single photographs. In this paper, we examine a novel way to denoise photos using only select regions from a burst of snapshots. First, we train an SVM to recognize image regions that stand to benefit the most from burst denoising. Then, whenever a burst of images is taken, only those regions selected by the SVM as 'beneficial' are stored. On a programmable camera, such as the Frankencamera, these regions can be selected and stored automatically. Finally, a non-local means denoising algorithm is performed offline, where the select regions are leveraged to improve the original image. The end result is noise reduction comparable to burst denoising, without the associated storage cost.

1. Introduction

An equation for signal to noise ratio in digital cameras is given below:

$$SNR = \frac{PQ_e t}{\sqrt{PQ_e t + Dt + N_r^2}},$$

where P is the number of photons per second, Q_e is the quantum efficiency, t is the exposure time, D is the dark current noise, and N_r is the read noise. Noise that is constant across images is not shown in the equation because it can be calculated and removed easily.

The above equation shows that increasing the exposure time will increase the signal to noise ratio. However, it is not always possible to increase the exposure time. For example, increasing the exposure time of a handheld camera can result in a blurry image due to camera shake. Instead, the photographer can take a burst of short exposure images, align them, and average them together. Since averaging together several exposures is effectively increasing the exposure time, the equation shows that this will increase the signal to noise ratio.

While this avoids the pitfalls of simply increasing exposure time, aligning and averaging entire images can introduce motion blur and ghosting effects. Moreover, it is expensive to store an entire burst of photos.

In this paper, we introduce a new technique which stores a representative subset of image regions from the entire burst of photos. Because we are using small regions instead of entire photos, this will reduce ghosting effects caused by motion across the images. Also, by storing only a subset of image patches, we address the problem of memory usage.

We use an SVM to determine which patches are the most important to store, and collect patches from different regions of the image to encourage variety in the patches. These patches can then be used to perform non-local means denoising.

2. Previous work

The bilateral filter [5] can be used to denoise images while preserving edges. This approach, however, still has the disadvantage of performing spatial filtering in a neighborhood, which results in textures being smoothed. This results in an undesirable loss of texture, and leads to poor results around edges.

Another simple denoising technique is non-local means (NL-means) [2], which instead averages together pixels with similar surrounding regions. The proximity of the pixels is not taken into account at all. This is an effective technique because it reflects the fact that repeated patterns are often found in separate regions of an image, and thus similar pixels may not be in the same immediate neighborhood. NL-means is also more effective at improving noisy pixels, since an entire neighborhood is used to determine pixel similarity, rather than just a single noisy pixel value.

Burst denoising and video denoising methods, such as [4] and [1], are also common. One advantage of these methods is that temporal filtering is used, which reduces potential blurring. When the images in the burst are properly aligned, these techniques effectively increase the exposure time in the aforementioned SNR equation. While these techniques deal with motion blur and ghosting artifacts, they are expensive because of the need to store an entire burst of photos.

In addition to these traditional approaches to deniosing, there are some existing techniques which explicitly use machine learning to denoise photos. One of them, by Yang *et al.* [7], trains an ϵ-SVM to approximate a bilateral filter.

Our approach leverages the predictive power of machine learning to supplement the aforementioned NL-means algorithm, and applies the NL-means algorithm in a novel way.

2.0.1 Bilateral filter in detail

The bilateral filter formula is given below:

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S_{\mathbf{p}}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, G_{\sigma_r}(\|I_{\mathbf{p}} - I_{\mathbf{q}}\|)\, I_{\mathbf{q}},$$

where G is the Gaussian function

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

$\mathbf{p}$ and $\mathbf{q}$ are x, y pixel coordinates and $I_{\mathbf{a}}$ is the pixel value at pixel coordinate $\mathbf{a}$. The Gaussian functions G_{σ_s} and G_{σ_r} provide the weights for the weighted average of the set $S_{\mathbf{p}}$ of pixels $I_{\mathbf{q}}$ surrounding pixel $I_{\mathbf{p}}$. $W_{\mathbf{p}}$ is the sum of the Gaussian weights calculated for each pixel in $S_{\mathbf{p}}$ ($\frac{1}{W_{\mathbf{p}}}$ is the normalizing term).

Again, the bilateral filter is useful in that it can reduce noise while preserving edges to some degree. However, because it performs spatial filtering, there will be some blurring of more complex regions - namely, textures and edges. By training our SVM to determine the difference between the bilateral filtered and high-quality images, we can isolate these complex regions and store them for denoising.

3. Methodology

3.1. Obtain training data

First, we obtained an image pair consisting of a noisy image and a high-quality image of the same scene. To do this, we used a tripod to take a burst of high-ISO photos, then averaged these images together. A single high-ISO image serves as our noisy image, and the averaged image serves as our high-quality image.

We then perform bilateral filtering on the noisy image, with σ_s=2, σ_r=0.045 (see Figure 1). We subtract the bilateral filtered image from the high-quality image and square the result, so that we get an absolute measure of how far off the bilateral filtered estimate is from the high-quality photo. We call this the ‘difference image’ (see Figure 2). We will want to train our SVM to predict ’difference images’.

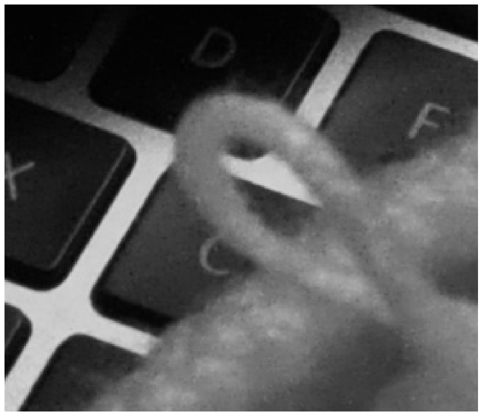
3.2. Train an SVM to find ‘important patches’

We define an important patch as an $n \times n$ region that cannot easily be denoised using a simple bilateral filter.

For each pixel, we obtain an n^2-dimensional vector, comprised of the pixels values of the surrounding $n \times n$ region. This n^2-dimensional feature vector is then mapped to a value z, which is the corresponding pixel in the difference image. To perform this mapping, we learn a mapping function using ϵ-Support Vector Regression [6]. Given a set of m training examples, $\{(\mathbf{x_1}, z_1), ..., (\mathbf{x_m}, z_m)\}$, where $\mathbf{x_i} \in \mathbb{R}^{(n^2)}$ is a feature vector and $z_i \in \mathbb{R}$ is the corresponding target variable, training our ϵ-SVM requires solving the following optimization problem:

$$\begin{array}{cc} \min\limits_{\mathbf{w},b,\zeta,\zeta^*} & 0.5\mathbf{w}^T\mathbf{w} + C\sum_{p=1}^{l}(\zeta_p + \zeta_p^*) \\ \text{s.t.} & \mathbf{w}^T\phi(\mathbf{x}_p) + b - z_p \le \epsilon + \zeta_p \\ & z_p - \mathbf{w}^T\phi(\mathbf{x}_p) - b \le \epsilon + \zeta_p^* \\ & \zeta_p, \zeta_p^* \ge 0, p = 1, ..., l \end{array}$$

Predictions within ϵ of the true value are not penalized. C is a constant term penalty for predictions that are not within ϵ of the true value, and ζ_p, ζ_p^* are slack variables that control the upper error bound. We used the ϵ-Support Vector Regression provided with SVM-LIB [3], and a simple linear kernel, for speed purposes. We ran 10-fold cross validation multiple times to select the appropriate parameters, and the parameter values we decided on were $C = 5, \epsilon = .1$.



1: Noisy input image

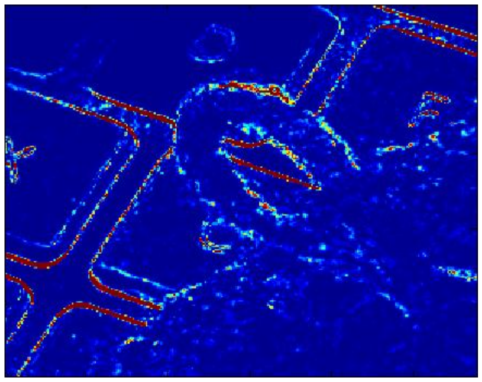


Figure 2: Difference image (red = high, blue = low)

3.3. Use the SVM: take a burst of images, and store the important patches

Take a burst of images. The first image will be stored in its entirety. We call this the this the ‘base image’, and the other images the ‘support images’. Our trained SVM will examine each support image and determine some X important patches for each image. These patches will be stored, and the support images will be discarded.

To encourage variety in the X patches, we split each support image into k subregions, such that the top X important patches are distributed evenly among the subregions. For our tests, we split up the images into 3×3 subregions.

It would be expensive to have the SVM make a prediction on each patch in each region, so we instead choose patches at random within each of the k subregions. Our SVM then determines the top X/k most important patches it encounters in each of these subregions. These regions will be stored; the rest will be discarded.

3.4. Piece together the final denoised image using NL-means

We now have the base image and X important patches. We will employ this collection of important patches to denoise our base image using a modified NL-means algorithm. For each pixel p in the base image, we construct a vector $\mathbf{v}_p \in \mathbb{R}^{(n^2)}$. We compare these vectors to all patches in the set S, which consists of surrounding patches in the base image and the set of ‘important patches’ determined in the previous set. The pixel is then set to be the weighed average of pixels with patches similar to its own. This calculation is performed offline.

The formula for the weighted average is as follows:

$$R_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{x} \in S} G_{\sigma_r}(\|\mathbf{v}_{\mathbf{p}} - \mathbf{x}\|^2)\, \mathbf{x}$$

4. SVM Results

We trained our SVM on the 288×248 image depicted in Figure 1. Our features were the 3×3 patches that surround each of the pixels, and our target values were the corresponding values in the ‘difference image’, depicted in Figure 2. Again, our SVM was trained using using $C = 5$ and $\epsilon = .1$.

Below are the results of SVM predictions, along with the ground zero truth images - that is, the actually ‘difference images’. Each of these images is individually scaled to better show the relative importance of each image region. The ‘zipper’ image was 288×248 pixels, the ‘text’ image was 298×228 pixels, and the ‘face’ image was 1278×1308 pixels.

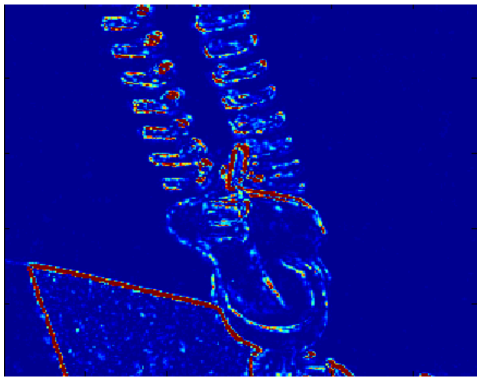


Figure 3: Zipper ground truth difference image

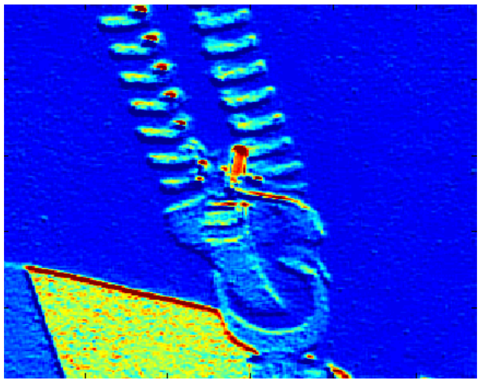


Figure 4: Zipper SVM difference image

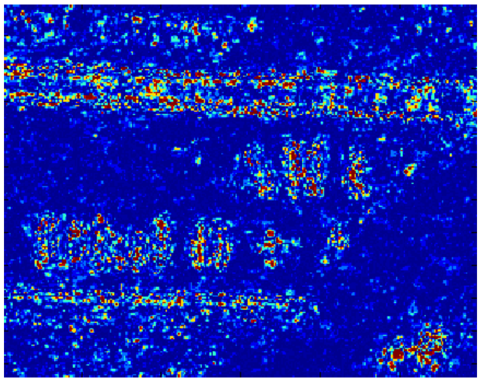


Figure 5: Text ground truth difference image

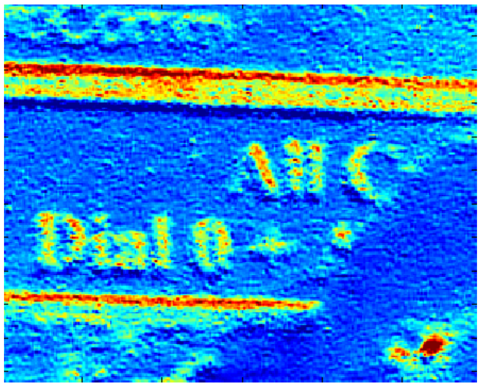


Figure 6: Text SVM difference image

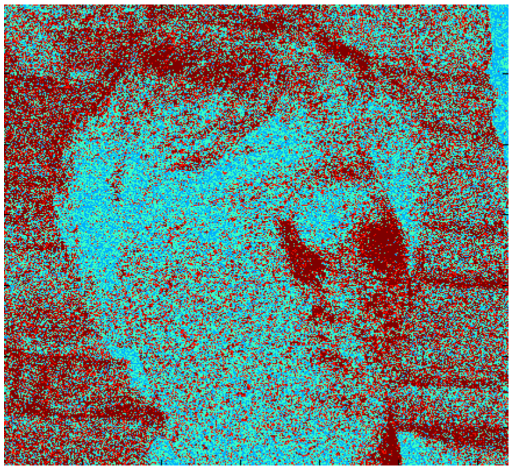


Figure 7: Face ground truth difference image

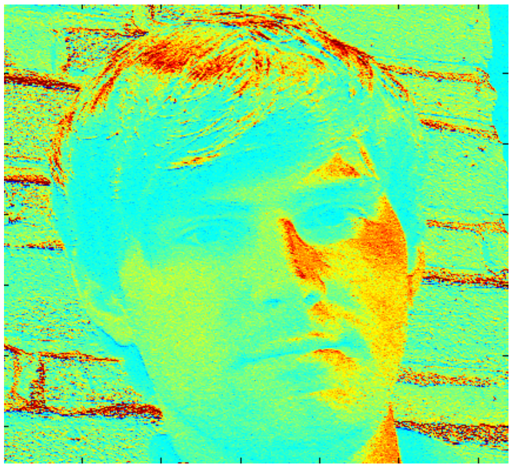


Figure 8: Face SVM difference image

We thought that these predictions looked quite good, especially given the small size of the training image. Our SVM was especially good at recognizing edge regions, almost all of which were important. We would have liked more importance to be placed on textures, but our difference images do not weigh textured regions very heavily. Also, we had to use a linear kernel since running higher dimensional kernels took prohibitively long. In light of this, we were surprised by the effectiveness of the SVM.

5. Denoising results

Here is our denoising result for the zipper image. We used 13 support images, and our algorithm determined a subset of the important regions which took up roughly one picture's worth of storage. For the NL-means, we used $\sigma = 0.03$.

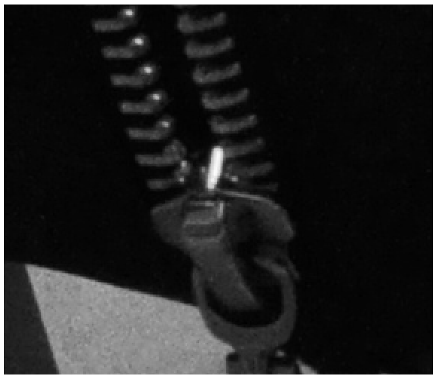


Figure 9: Noisy base image

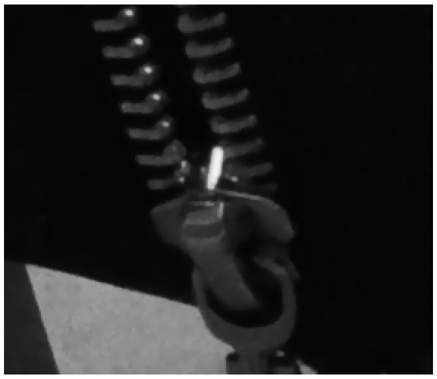


Figure 10: Bilateral filtered image

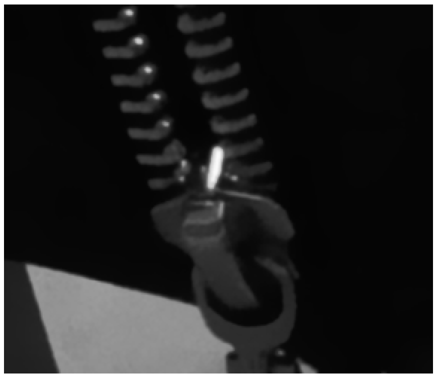


Figure 11: Our result

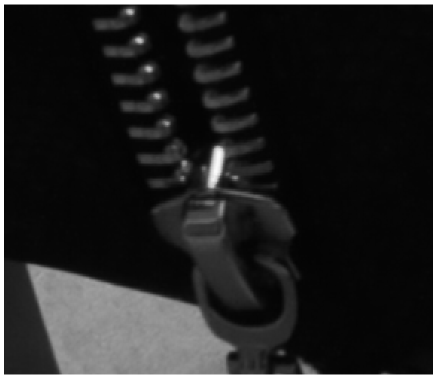


Figure 12: High-quality image

6. Discussion and future work

We thought that our result looked noticeably sharper than the bilateral filtered image. It clearly performs less blurring across edges. However, when compared with a simple NL-means algorithm on the base image, our results were only very slightly better in terms of mean squared error. There were slight or no visual difference. This leads us to conclude that, because NL-means is already an effective offline denoising algorithm, the additional regions detected by our SVM are not necessary for offline denoising. We hypothesize that the extra regions may be helpful for a scene that contains many unique regions. Since our images had repeated patterns, however, we were not able to test this case.

Although our regions were not very helpful for offline NL-means denoising, they may be more useful for online denoising. If we store these patches in a kd-tree or other data structure that allows quick searching for similar patches, it may be possible to denoise images on the fly in the camera, using only important regions, since running NL-means on a camera would be too expensive.

In summary, we found that our trained SVM was very effective at recognizing those regions of an image for which bilateral filtering is inefective. Using these extra regions to perform NL-means denoising significantly reduced noise, and our approach noticeably outperformed the bilateral filer. However, our results were very similar to those obtained when performing NL-means on the base image alone, which already closely approximates an aligned and averaged burst of snapshots. We conclude that providing the NL-means with additional image regions from a burst of photos is mostly unnecessary. Nonetheless, the effectiveness of the SVM at predicting important edge and texture regions is noteworthy, and while using these regions to assist an NL-means algorithm proved unnecessary, this application of SVM's might be relevant to related problems in computational photography.

References

[1] E. P. Bennett and L. McMillan. Video enhancement using per-pixel virtual exposures. *ACM SIGGRAPH 2005 Papers*, 2005.

[2] A. Buades, B. Coll, and J. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal*, 4:490–530, 2005.

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[4] H. J. Seo and P. Milanfar. Video denoising using higher order optimal space-time adaptation. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008.

[5] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision*, 1998.

[6] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[7] Q. Yang, S. Wang, and N. Ahuja. Svm for edge-preserving filtering. In *CVPR*, 2010.