Variance Stabilization for Noisy+Estimate Combination in Iterative Poisson Denoising

Published in
IEEE Signal Processing Letters

DOI
10.1109/LSP.2016.2580600

Variance Stabilization for Noisy+Estimate Combination in Iterative Poisson Denoising

Lucio Azzari and Alessandro Foi

***Abstract*—We denoise Poisson images with an iterative algorithm that progressively improves the effectiveness of variance-stabilizing transformations (VST) for Gaussian denoising filters. At each iteration, a combination of the Poisson observations with the denoised estimate from the previous iteration is treated as scaled Poisson data and filtered through a VST scheme. Due to the slight mismatch between a true scaled Poisson distribution and this combination, a special exact unbiased inverse is designed. We present an implementation of this approach based on the BM3D Gaussian denoising filter. With a computational cost at worst twice that of the non-iterative scheme, the proposed algorithm provides significantly better quality, particularly at low SNR, outperforming much costlier state-of-the-art alternatives.**

***Index Terms*—image denoising, photon-limited imaging, Poisson noise, Anscombe transformation, iterative filtering.**

I. Introduction

Denoising of images affected by Poisson noise is commonly executed by: 1) applying a variance stabilizing transformation (VST) to standardize the image noise, 2) denoising the image with an additive white Gaussian noise (AWGN) filter, 3) returning the image to its original range via inverse transformation. The most common VST for this purpose is the Anscombe transformation [1], [2]. Since, at very low counts (*e.g.*, less than one count per pixel, with SNR≪0dB), the Anscombe transformation can be quite inaccurate [3], denoising algorithms specifically designed for Poisson noise [3], [4] were developed to provide better performances than combinations of VST with Gaussian filters.

However, being inexpensive, simple, and independent from the adopted denoising algorithm, the Anscombe transform is still very appealing, and in this letter we propose an iterative algorithm based on the VST framework that is capable of dealing with challenging cases with extremely low SNR, and that outperforms state-of-the-art algorithms, both in terms of image quality and execution time.

At each step we apply the VST approach to a combination of the initial observed image and its most recent estimateto increase the SNR of the signal to be denoised, and consequently to improve the effectiveness of the stabilization and filtering. We analyze the statistics of this combination, which deviate from a Poisson distribution, and introduce the corresponding exact unbiased inverse to be used in this VST framework. We present an implementation of this approach based on the BM3D Gaussian denoising filter [5]. With a computational cost at worst twice that of the non-iterative scheme, the proposed algorithm provides significantly better quality, particularly at low and extremely low SNR, outperforming much costlier state-of-the-art alternatives.

This work was supported by the Academy of Finland (project no. 252547).
The authors are with the Department of Signal Processing, Tampere University of Technology, Tampere, FI-33101, Finland (e-mail: lucio.azzari@tut.fi; alessandro.foi@tut.fi).

II. Preliminaries and motivation

Let z be an observed noisy image composed of pixels $z(x)$, $x\in\Omega\subset\mathbb{Z}^2$, modeled as independent realizations of a Poisson process with parameter $y(x)\geq 0$:

$$z(x)\sim\mathcal{P}(y(x)),\quad \mathrm{P}(z(x)\,|y(x))=\begin{cases}\frac{y(x)^{z(x)}e^{-y(x)}}{z(x)!} & z\in\mathbb{N}\cup\{0\}\\ 0 & \text{elsewhere.}\end{cases}$$

The mean and variance of $z(x)$ coincide and are equal to $y(x)$:

$$\mathrm{E}\{z(x)\,|y(x)\}=\mathrm{var}\{z(x)\,|y(x)\}=y(x)\,.$$

For conciseness, henceforth we will omit x from notation.

Our goal is to compute an estimate $\hat{y}$ of y from z. To this purpose, in the archetypal VST framework, the Anscombe forward transformation a [1] yields an image

$$a(z)=2\sqrt{z+3/8}$$

which can be treated as corrupted by AWGN with unit variance. Thus, it can be denoised using any filter Φ designed for AWGN. If the denoising is ideal, we have

$$\Phi\left[a(z)\right]=\mathrm{E}\{a(z)\,|y\}\,.$$

The so-called *exact unbiased inverse* of a [2]

$$\mathcal{I}_a^{\mathcal{P}}:\mathrm{E}\{a(z)\,|y\}\mapsto\mathrm{E}\{z|y\}=y,$$

is used to return the denoised image to the original range of z, thus yielding an estimate of y:

$$\hat{y}=\mathcal{I}_a^{\mathcal{P}}\left(\Phi\left[a(z)\right]\right).$$

However, for small y, when the SNR is very low, the stabilization is imprecise and the conditional distribution of $a(z)$ is far from the assumed normal, in terms of both scale and shape, leading to ineffective filtering with Φ. This issue has been commonly addressed either by applying VST after binning, i.e. by stabilizing sums of adjacent pixels instead of individual pixels [3], [4], [6]–[10], or by similarly stabilizing transform coefficients [11] (essentially inserting the VST within the denoising method itself). All these stratagems aim at increasing the SNR of the data subject to the VST.

In this letter, we introduce an alternative and more direct way to improve the SNR prior to VST, by combining the noisy observation z with a previously obtained estimate of the noise-free data y, leading to the following simple iterative algorithm.

III. Proposed iterative algorithm

A subscript index denotes a symbol's instance at a particular iteration, *e.g.*, $\hat{y}_i$ is the estimate of y at iteration i.

We initialize the algorithm by setting $\hat{y}_0 = z$. At each iteration $i = 1, \dots, K$ we compute a convex combination of $\hat{y}_{i-1}$ and z

$$\bar{z}_i = \lambda_i z + (1 - \lambda_i)\, \hat{y}_{i-1}, \tag{1}$$

where $0 < \lambda_i \leq 1$. Provided we can treat $\hat{y}_{i-1}$ as a surrogate for y, we have $\mathrm{E}\{\bar{z}_i|y\} = y = \lambda_i^{-2}\mathrm{var}\{\bar{z}_i|y\}$; thus $\bar{z}_i$ has higher SNR than z for any $\lambda_i < 1$. We then apply a VST f_i to $\bar{z}_i$ and obtain an image $\bar{\bar{z}}_i = f_i(\bar{z}_i)$, which we denoise with a filter Φ for AWGN to obtain a filtered image $D_i = \Phi\left[\bar{\bar{z}}_i\right]$. Assuming $D_i = \mathrm{E}\{f_i(\bar{z}_i)|y\}$, the exact unbiased inverse of f_i, $\mathcal{I}_{f_i}^{\lambda_i} : \mathrm{E}\{f_i(\bar{z}_i)|y\} \mapsto \mathrm{E}\{\bar{z}_i|y\} = y$, brings this image to the original range, yielding

$$\hat{y}_i = \mathcal{I}_{f_i}^{\lambda_i}(D_i)\,,$$

which is either used for the next iteration if $i < K$, or output as final estimate $\hat{y}_K = \hat{y}$.

Let us provide further details on the above basic procedure.

A. Forward variance-stabilizing transformation

Consider the scaled variable $\lambda_i^{-2}\bar{z}_i$ and let us model $\hat{y}_{i-1}$ as y. Setting $q_i(t) = \lambda_i t - \frac{1-\lambda_i}{\lambda_i} y$, the conditional probability

$$\mathrm{P}\left(\lambda_i^{-2}\bar{z}_i|y\right) = \begin{cases} \frac{y^{q_i(\lambda_i^{-2}\bar{z}_i)}e^{-y}}{q_i(\lambda_i^{-2}\bar{z}_i)!} & q_i\left(\lambda_i^{-2}\bar{z}_i\right) \in \mathbb{N}\cup\{0\} \\ 0 & \text{elsewhere}\,. \end{cases} \tag{2}$$

Unless $\lambda_i = 1$, this is *not* a Poisson distribution. However, the mean and variance of $\lambda_i^{-2}\bar{z}_i$ do always coincide:

$$\mathrm{E}\left\{\lambda_i^{-2}\bar{z}_i\middle|y\right\} = \mathrm{var}\left\{\lambda_i^{-2}\bar{z}_i\middle|y\right\} = \lambda_i^{-2}y.$$

Hence, $\lambda_i^{-2}\bar{z}_i$ resembles $\mathcal{P}\left(\lambda^{-2}y\right)$ and indeed one can prove [12] that it is asymptotically stabilized by the Anscombe transformation a. Thus, we set $f_i(\cdot) = a\left(\lambda_i^{-2}(\cdot)\right)$.

B. Exact unbiased inverse transformation

The exact unbiased inverse $\mathcal{I}_{f_i}^{\lambda_i}$ is defined upon (2) as

$$\mathrm{E}\{f_i(\bar{z}_i)|y\} = \sum_{\bar{z}_i : q_i\left(\lambda_i^{-2}\bar{z}_i\right) \in \mathbb{N}\cup\{0\}} a\left(\lambda_i^{-2}\bar{z}_i\right) \mathrm{P}\left(\lambda_i^{-2}\bar{z}_i\middle|y\right) \mapsto \mathrm{E}\{\bar{z}_i|y\} = y. \tag{3}$$

We have $\mathcal{I}_{f_i}^{\lambda_i} \approx \lambda_i^2 \mathcal{I}_a^{\mathcal{P}}$, with $\mathcal{I}_{f_i}^{1} = \mathcal{I}_a^{\mathcal{P}}$ [2]. The appendix describes how to accurately compute (3) in practice.

C. Binning

It is natural to combine the convex combination (1) with a linear binning; this can be especially beneficial at the first iterations, when $\hat{y}_{i-1}$ is a poor estimate of y. Specifically, a binning operator $\mathcal{B}_{h_i}$ can be applied to $\bar{z}_i$, yielding a smaller image where each block (*i.e.* bin) of $h_i \times h_i$ pixels from $\bar{z}_i$ is replaced by a single pixel equal to their sum. $\mathcal{B}_{h_i}$ clearly commutes with (1) and

$$\mathcal{B}_{h_i}[\bar{z}_i] = \lambda_i \mathcal{B}_{h_i}[z] + (1 - \lambda_i)\, \mathcal{B}_{h_i}[\hat{y}_{i-1}]\,.$$

Algorithm 1 Iterative Poisson Image Denoising via VST

1: $\hat{y}_0 = z$
2: **for** $i = 1$ to K **do**
3: $\quad \bar{z}_i = \lambda_i z + (1 - \lambda_i)\hat{y}_{i-1}$
4: $\quad \bar{\bar{z}}_i = f_i\left(\mathcal{B}_{h_i}[\bar{z}_i]\right)$
5: $\quad D_i = \Phi\left[\bar{\bar{z}}_i\right]$
6: $\quad \hat{y}_i = \mathcal{B}_{h_i}^{-1}\left[\mathcal{I}_{f_i}^{\lambda_i}(D_i)\right]$
7: **end for**
8: **return** $\hat{y} = \hat{y}_K$

Algorithm 2 Debinning $\hat{y}_i = \mathcal{B}_{h_i}^{-1}\left[\mathcal{I}_{f_i}^{\lambda_i}(D_i)\right]$

1: $\hat{y}_{i,0} = 0$
2: **for** $j = 1$ to J **do**
3: $\quad r_j = \mathcal{I}_{f_i}^{\lambda_i}(D_i) - \mathcal{B}_{h_i}\left[\hat{y}_{i,j-1}\right]$
4: $\quad \hat{y}_{i,j} = \max\left\{0,\, \hat{y}_{i,j-1} + \mathcal{U}_{h_i}\left[h^{-2}r_j\right]\right\}$
5: **end for**
6: **return** $\hat{y}_i = \hat{y}_{i,J}$

Since $\mathcal{B}_{h_i}[z] \sim \mathcal{P}\left(\mathcal{B}_{h_i}[y]\right) = \mathcal{P}\left(\mathrm{E}\{\mathcal{B}_{h_i}[z]|y\}\right)$, and modeling again $\hat{y}_{i-1}$ as y, we have that $\mathcal{B}_{h_i}[\bar{z}_i]$ (resp. $\lambda_i^{-2}\mathcal{B}_{h_i}[\bar{z}_i]$) is subject to the same conditional probability of $\bar{z}_i$ (resp. $\lambda_i^{-2}\bar{z}_i$), which means that the adoption of binning does not interfere with the subsequent VST, denoising, and inverse VST. Thus, we can define $\bar{\bar{z}}_i = f_i\left(\mathcal{B}_{h_i}[\bar{z}_i]\right)$ without modifying f_i .

Debinning: An inverse binning operator $\mathcal{B}_{h_i}^{-1}$ is applied after the exact unbiased inversion,

$$\hat{y}_i = \mathcal{B}_{h_i}^{-1}\left[\mathcal{I}_{f_i}^{\lambda_i}(D_i)\right],$$

returning a full-size image estimate $\hat{y}_i$ such that

$$\mathcal{B}_{h_i}[\hat{y}_i] = \mathcal{I}_{f_i}^{\lambda_i}(D_i)\,. \tag{4}$$

All the above steps are summarized in Algorithm 1 and as

$$\hat{y}_i = \mathcal{B}_{h_i}^{-1}\left[\mathcal{I}_{f_i}^{\lambda_i}\left(\Phi\left[f_i\left(\mathcal{B}_{h_i}\left[\lambda_i z + (1 - \lambda_i)\hat{y}_{i-1}\right]\right)\right]\right)\right].$$

IV. Implementation and Results

For Φ we adopt the BM3D denoising algorithm [5]; yet, other AWGN filters such as [13]–[20] may be used as well, and also lead to competitive results as shown in the supplementary material [21].

In the debinning step, to compute $\mathcal{B}_{h_i}^{-1}[\mathcal{I}_{f_i}^{\lambda_i}(D_i)]$, $\mathcal{I}_{f_i}^{\lambda_i}(D_i)$ is first divided by h_i^2, *i.e.* by number of pixels in the bin, and upscaled to the size of z via cubic spline interpolation $\mathcal{U}_{h_i}$. To enforce the constraint (4), the output of interpolation is recursively binned by $\mathcal{B}_{h_i}$ and subtracted from the target $\mathcal{I}_{f_i}^{\lambda_i}(D_i)$, giving a residual which is upsampled and accumulated. This subroutine, summarized in Algorithm 2, is an instance of the recursive shaping regularization with nonnegativity [22], [23].

Our current implementation[1] of Algorithm 1 is determined by four parameters: K (number of iterations), λ_K, h_1, h_K (first and last bin sizes); other values of λ_i and h_i are defined as $\lambda_i = 1 - \frac{i-1}{K-1}(1-\lambda_K)$ and $h_i = \max\{h_K, h_1 - 2i + 2\}$. We use

[1] Matlab software available at http://www.cs.tut.fi/~foi/invansc/

Table I
PSNR (dB) DENOISING RESULTS VS [2]–[4], [7], WITH AND WITHOUT 3×3 BINNING. AVERAGES OVER 5 NOISE REALIZATIONS. P^{4}IP VALUES FROM [7].

Method	Peak	Flag$_{256^2}$	House$_{256^2}$	Cam$_{256^2}$	Man$_{512^2}$	Bridge$_{256^2}$	Saturn$_{256^2}$	Peppers$_{256^2}$	Boat$_{512^2}$	Couple$_{512^2}$	Hill$_{512^2}$	Time$_{256^2}$
NLSPCA [3]	0.1	14.60	17.63	16.63	18.35	16.71	20.45	16.17	18.14	18.49	18.80	89s
NLSPCA bin [3]		15.32	**18.66**	17.23	18.41	16.99	18.91	16.22	18.89	18.84	19.47	11s
SPDA [4]		13.51	14.58	14.34	–	14.67	17.57	14.34	–	–	–	8h
SPDA bin [4]		15.27	18.29	16.83	18.72	17.00	21.53	16.15	18.99	18.94	19.39	12min
P^{4}IP [7]		13.30	18.30	16.88	–	16.45	21.55	16.28	–	–	–	few mins
VST+BM3D [2]		12.38	15.69	15.44	16.95	15.60	18.40	15.15	16.22	16.50	16.79	0.70s
VST+BM3D bin [2]		13.97	18.22	16.99	18.61	16.93	20.09	15.84	18.91	18.62	19.23	0.11s
Proposed		**16.01**	18.48	**17.45**	**18.96**	**17.30**	**21.64**	**16.45**	**19.32**	**19.31**	**19.68**	0.48s
NLSPCA [3]	0.2	16.47	18.63	17.63	19.18	17.56	21.36	17.21	19.14	19.22	19.74	90s
NLSPCA bin [3]		15.63	19.21	17.87	19.12	17.40	19.67	16.69	19.48	19.37	19.99	12s
SPDA [4]		16.65	17.45	16.75	–	16.96	20.67	16.70	–	–	–	5h
SPDA bin [4]		17.41	18.95	17.80	19.73	17.81	22.90	17.25	19.85	19.72	20.36	27min
P^{4}IP [7]		14.82	19.48	17.82	–	17.54	23.05	17.31	–	–	–	few mins
P^{4}IP bin [7]		17.26	**19.96**	**18.58**	–	17.54	**23.79**	17.44	–	–	–	~30s
VST+BM3D [2]		13.53	17.79	16.90	18.69	17.12	21.38	16.96	18.23	18.47	18.80	0.69s
VST+BM3D bin [2]		16.85	19.27	17.88	19.82	17.70	22.94	17.19	19.79	19.71	20.09	0.12s
Proposed		**17.49**	19.68	18.40	**19.94**	**18.13**	23.13	**17.54**	**20.09**	**20.03**	**20.48**	0.83s
NLSPCA [3]	0.5	18.61	20.17	19.20	20.59	18.49	22.89	18.69	20.37	20.42	21.14	96s
NLSPCA bin [3]		15.76	20.48	18.26	19.77	18.17	21.65	17.69	20.11	20.01	20.67	19s
SPDA [4]		**20.02**	19.96	18.75	–	18.52	25.37	18.55	–	–	–	4h
SPDA bin [4]		18.40	20.57	18.87	20.70	18.57	**25.93**	18.52	20.84	20.70	21.35	23min
P^{4}IP [7]		16.50	20.93	19.27	–	18.47	25.19	18.86	–	–	–	few mins
VST+BM3D [2]		15.58	19.61	18.46	20.39	18.26	23.75	18.41	19.99	20.01	20.74	0.71s
VST+BM3D bin [2]		18.19	21.41	19.47	21.15	18.71	25.81	18.78	20.94	20.83	21.72	0.11s
Proposed		18.60	**21.54**	**19.79**	**21.25**	**19.08**	25.78	**19.05**	**21.19**	**21.14**	**21.84**	0.83s
NLSPCA [3]	1	19.68	21.57	20.25	21.46	19.02	24.75	19.50	21.19	21.14	21.94	86s
NLSPCA bin [3]		15.77	20.78	18.40	19.87	18.26	22.83	17.78	20.19	20.11	20.82	16s
SPDA [4]		**22.97**	22.14	20.15	–	19.30	27.05	19.97	–	–	–	5h
SPDA bin [4]		18.99	20.99	19.43	21.15	18.84	27.40	18.93	21.19	20.97	21.50	25min
P^{4}IP [7]		19.07	22.67	20.54	–	19.31	27.05	20.07	–	–	–	few mins
VST+BM3D [2]		18.46	21.64	20.19	21.62	19.43	25.82	19.71	21.47	21.14	21.92	0.78s
VST+BM3D bin [2]		19.28	22.53	20.69	22.07	19.59	**27.59**	20.22	21.97	21.81	22.72	0.10s
Proposed		19.74	**23.04**	**21.07**	**22.30**	**19.86**	27.27	**20.44**	**22.17**	**22.08**	**22.85**	0.82s
NLSPCA [3]	2	19.70	23.16	20.64	22.37	19.43	26.88	20.48	21.83	21.75	22.68	87s
NLSPCA bin [3]		15.52	20.85	18.35	19.87	18.32	21.27	17.78	20.29	20.21	20.98	12s
SPDA [4]		**24.72**	24.37	21.35	–	20.17	29.13	21.18	–	–	–	6h
SPDA bin [4]		19.26	21.12	19.53	21.66	18.87	28.54	19.17	21.43	21.24	21.94	25min
P^{4}IP [7]		21.04	**24.65**	21.87	–	20.16	28.93	21.33	–	–	–	few mins
VST+BM3D [2]		20.79	23.79	21.97	23.11	20.49	27.95	**22.02**	22.90	22.65	23.34	0.82s
VST+BM3D bin [2]		19.91	24.10	21.43	23.03	20.36	**29.26**	21.45	22.92	22.84	23.75	0.10s
Proposed		21.18	24.62	**22.25**	**23.40**	**20.69**	28.85	21.93	**23.30**	**23.12**	**23.88**	0.82s
NLSPCA [3]	4	20.15	24.26	20.97	22.93	20.21	27.99	21.07	22.49	22.33	23.51	123s
NLSPCA bin [3]		15.52	20.94	18.27	19.88	18.32	22.02	17.72	20.29	20.25	20.99	13s
SPDA [4]		**25.76**	25.30	21.72	–	20.53	**31.13**	22.20	–	–	–	8h
SPDA bin [4]		19.42	22.07	19.95	22.18	19.26	29.71	20.19	21.76	21.69	22.82	31min
P^{4}IP [7]		22.49	**26.33**	23.29	–	21.11	30.82	23.88	–	–	–	few mins
VST+BM3D [2]		22.93	25.49	23.82	24.32	21.51	29.41	24.01	24.16	24.10	24.47	0.74s
VST+BM3D bin [2]		20.43	25.49	22.22	23.99	21.13	30.87	22.57	23.92	23.84	24.69	0.10s
Proposed		23.51	26.07	**24.10**	**24.52**	**21.71**	30.38	**24.04**	**24.53**	**24.34**	**24.82**	1.41s

decreasing h_i since binning can cause loss of image details and it becomes progressively less useful when λ_i gets larger and the role of $\hat{y}_{i-1}$ dominates in improving the SNR of the VST input. Obviously, $\mathcal{B}_1$ and $\mathcal{B}_1^{-1}$ are identity operators.

The Poisson image z is the only input to our algorithm; the parameters K, λ_K, h_1, h_K are adaptively selected based on the quantiles of z, following a training [21] over 6 images not included in the experiments test dataset; we fix $J = 9$.

PSNR (dB) results of the proposed algorithm and [2]–[4], [7], as well as their versions with binning, are reported in Table I. Table II gives a separate comparison with [8], over the different dataset of 256×256 downscaled images adopted by its authors. The tables demonstrate the superior overall performance of the proposed algorithm, also confirmed by visual inspection of the examples in Figure 1.

The complexity of Algorithm 1 is dominated by the filter Φ and possibly by the debinning operators $\mathcal{B}_{h_i}^{-1}$. The overall execution time depends especially on the number of iterations K and on h_K, which sets the size of the largest image to be filtered by Φ; all our results have $K \leq 4$. Table I and Table II report also the average execution times for 256×256 images. We ran the proposed algorithm and [2]–[4] on a single thread of a 3.4-GHz Intel i7 CPU; the runtimes for [7], [8] are taken from the respective articles, where [8] uses a 3.3-GHz Intel i7, and [7] also uses an Intel i7. The proposed algorithm and [2] are significantly less expensive than any of the other methods.

V. Discussion and Conclusions

We presented an iterative VST framework for Poisson denoising. The iterative combination with a previous estimate refines

Table II
PSNR (dB) DENOISING RESULTS VERSUS THOSE REPORTED IN [8]. AVERAGES OVER 5 NOISE REALIZATIONS.

Method	Peak	Peppers$_{256^2}$	Bridge$_{256^2}$	Boat$_{256^2}$	Couple$_{256^2}$	Hill$_{256^2}$	Mandrill$_{256^2}$	Man$_{256^2}$	Time$_{256^2}$
MMSE est [8]	1	20.38	19.55	20.24	20.26	20.98	18.43	20.49	~14min
Proposed		**20.44**	**19.86**	**20.65**	**20.47**	**21.23**	**18.56**	**20.50**	0.82s
MMSE est [8]	2	**22.26**	20.65	21.28	21.22	22.05	18.98	21.60	~14min
Proposed		21.93	**20.69**	**21.46**	**21.40**	**22.32**	**19.14**	**21.62**	0.82s
MMSE est [8]	4	23.92	21.60	22.32	22.26	23.23	19.56	**22.79**	~14min
Proposed		**24.04**	**21.71**	**22.53**	**22.52**	**23.29**	**19.65**	22.75	1.41s

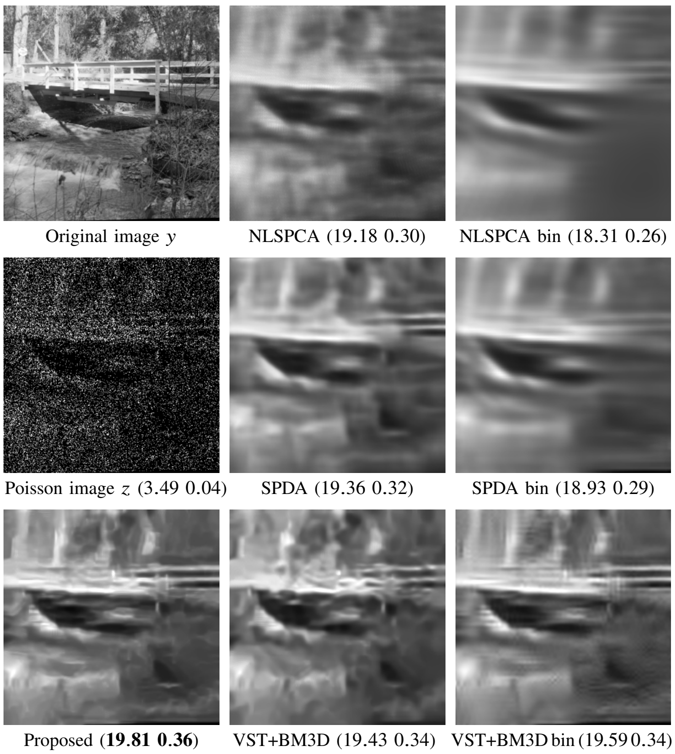

Figure 1. Denoising of *Bridge* at peak 1. PSNR (dB) and SSIM [24] of $\hat{y}$ are given in brackets. For clarity, z is visualized on a compressed range.

the stabilization and helps to cope with extreme low-SNR cases, in which a standard VST approach [2] underperforms even when endowed with binning.

To analyze the importance of embedding the VST framework within the iterations, in Table III we compare our results from Table I with those by a simplified version of Algorithm 1, where the VST is *external* to the loop: f_i and $\mathcal{I}_{f_i}^{\lambda_i}$ are replaced by identity operators and a and $\mathcal{I}_a^{\mathcal{P}}$ are applied outside of the algorithm. The significant gain in the table confirms that the improvement over [2] is not a mere consequence of a better denoising due to iterative filtering at multiple scales.

Also P^{4}IP [7] relies on iterative AWGN filtering to denoise the Poisson z. In contrast to P^{4}IP, which formulates an optimization problem to be solved upon convergence of ADMM [25] iterations, each iteration of Algorithm 1 attacks the Poisson denoising problem directly, so any $\hat{y}_k$ can be treated as an estimate of y, with $\hat{y}_1$ already coinciding with [2]. This results in a more efficient, stable, and substantially faster procedure, where Φ (*e.g.*, BM3D) is used as explicit denoiser for AWGN with variance 1 set by the VST without need of empirical tuning.

Table III
PSNR GAIN (AVERAGE OVER ALL IMAGES IN TABLE I) OF ALGORITHM 1 OVER ITS SIMPLIFICATION WITH *external* VST (SEE SECTION V).

Peak	0.1	0.2	0.5	1	2	4
PSNR (dB) gain	0.64	0.66	0.38	0.22	0.11	0.13

The proposed algorithm achieves state-of-the-art quality in only a tiny fraction of the time required by competitive algorithms.

APPENDIX: COMPUTING THE EXACT UNBIASED INVERSE

As in [2], we compute $\mathrm{E}\{a(\lambda_i^{-2}\bar{z}_i)\,|\,y\}$ numerically over a finite grid of values of y and λ_i, from which we interpolate $\mathcal{I}_{f_i}^{\lambda_i}$ (3) at values within the grid range. Outside of the grid range, we leverage the available implementation [2] of the exact unbiased inverse for Poisson $\mathcal{I}_a^{\mathcal{P}}:\mathrm{E}\{a(\lambda_i^{-2}\zeta)\,|\,y\}\mapsto\lambda_i^{-2}y$, where $\lambda_i^{-2}\zeta\sim\mathcal{P}(\lambda_i^{-2}y)$, through the composition

$$\mathrm{E}\{a(\lambda_i^{-2}\bar{z}_i)\,|\,y\}\longmapsto\mathrm{E}\{a(\lambda_i^{-2}\zeta)\,|\,y\}\longmapsto\lambda_i^{-2}y\longmapsto y. \qquad (5)$$

To deal with the first of the three mappings (5), we study the difference between $\mathrm{E}\{a(\lambda_i^{-2}\bar{z}_i)\,|\,y\}$ and $\mathrm{E}\{a(\lambda_i^{-2}\zeta)\,|\,y\}$. For $p\sim\mathcal{P}(\mu)$, the mean of a generic $g(p)=2\sqrt{(p+d)/\gamma}$ is [1], [26]

$$\mathrm{E}\{g(p)\,|\,\mu\}=2\sqrt{\frac{\mu+d}{\gamma}}\Big(1-\frac{1}{8}\frac{\mu}{(\mu+d)^2}+ \qquad (6)$$
$$+\frac{1}{16}\frac{\mu}{(\mu+d)^3}-\frac{5}{128}\frac{3\mu^2+\mu}{(\mu+d)^4}+O\big(\mu^{-3}\big)\Big).$$

It yields $\mathrm{E}\{a(\lambda_i^{-2}\bar{z}_i)\,|\,y\}$ when $\mu=y$, $\gamma=\lambda_i$, $d=\frac{1-\lambda_i}{\lambda_i}y+\frac{3}{8}\lambda_i$, and $\mathrm{E}\{a(\lambda_i^{-2}\zeta)\,|\,y\}$ when $\mu=\lambda^{-2}y$, $\gamma=1$, $d=\frac{3}{8}$. Then

$$\mathrm{E}\{a(\lambda_i^{-2}\zeta)\,|\,y\}-\mathrm{E}\{a(\lambda_i^{-2}\bar{z}_i)\,|\,y\}=\frac{\lambda_i^2(\lambda_i-1)}{8}y^{-\frac{3}{2}}+O\big(y^{-\frac{5}{2}}\big), \qquad (7)$$

which is however expressed as a function of y, while (5) requires a function of $\mathrm{E}\{a(\lambda^{-2}\bar{z})\,|\,y\}$. From (6), we can approximate large y as

$$y=\Big(\frac{\lambda_i}{2}\mathrm{E}\{a(\lambda_i^{-2}\bar{z}_i)\,|\,y\}\Big)^2+O(1)\,. \qquad (8)$$

On substituting (8) into (7) we obtain

$$\mathrm{E}\{a(\lambda_i^{-2}\zeta)\,|\,y\}-\mathrm{E}\{a(\lambda_i^{-2}\bar{z}_i)\,|\,y\}= \qquad (9)$$
$$=\frac{\lambda_i-1}{\lambda_i}\Big(\mathrm{E}\{a(\lambda_i^{-2}\bar{z}_i)\,|\,y\}\Big)^{-3}+O\Big(\mathrm{E}\{a(\lambda_i^{-2}\bar{z}_i)\,|\,y\}\Big)^{-4}.$$

Outside of the grid range we can discard the higher-order terms from (9) and compute $\mathcal{I}_{f_i}^{\lambda_i}(D_i)$ using $\mathcal{I}_a^{\mathcal{P}}$ [2] as

$$\mathcal{I}_{f_i}^{\lambda_i}(D_i)=\lambda_i^2\,\mathcal{I}_a^{\mathcal{P}}\Big(D_i+\frac{\lambda_i-1}{\lambda_i}D_i^{-3}\Big)\,.$$

References

[1] F. J. Anscombe, "The transformation of Poisson, binomial and negative-binomial data," *Biometrika*, vol. 35, no. 3/4, pp. 246–254, 1948. [Online]. Available: http://www.jstor.org/stable/2332343

[2] M. Mäkitalo and A. Foi, "Optimal inversion of the Anscombe transformation in low-count Poisson image denoising," *IEEE Transactions on Image Processing*, vol. 20, no. 1, pp. 99–109, Jan 2011, [Online]. Available: http://dx.doi.org/10.1109/TIP.2010.2056693. Matlab software at http://www.cs.tut.fi/~foi/invansc/.

[3] J. Salmon, Z. Harmany, C.-A. Deledalle, and R. Willett, "Poisson noise reduction with non-local PCA," *Journal of Mathematical Imaging and Vision*, vol. 48, no. 2, pp. 279–294, 2014, [Online]. Available: http://dx.doi.org/10.1007/s10851-013-0435-6. Software at http://josephsalmon.eu/.

[4] R. Giryes and M. Elad, "Sparsity-based Poisson denoising with dictionary learning," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5057–5069, Dec 2014, [Online]. Available: http://dx.doi.org/10.1109/TIP.2014.2362057. Software at http://tam-son3.eng.tau.ac.il/~raja/?page_id=32.

[5] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, Aug 2007, [Online]. Available: http://dx.doi.org/10.1109/TIP.2007.901238. Matlab software at http://www.cs.tut.fi/~foi/GCF-BM3D/.

[6] L. Brown, T. Cai, R. Zhang, L. Zhao, and H. Zhou, "The root–unroot algorithm for density estimation as implemented via wavelet block thresholding," *Probability Theory and Related Fields*, vol. 146, no. 3-4, pp. 401–433, 2010. [Online]. Available: http://dx.doi.org/10.1007/s00440-008-0194-2

[7] A. Rond, R. Giryes, and M. Elad, "Poisson inverse problems by the plug-and-play scheme," *arXiv e-print*, 2015. [Online]. Available: http://arxiv.org/abs/1511.02500

[8] S. Pyatykh and J. Hesser, "MMSE estimation for Poisson noise removal in images," *arXiv e-print*, 2015. [Online]. Available: http://arxiv.org/abs/1512.00717

[9] T. Remez, O. Litany, and A. Bronstein, "A picture is worth a billion bits: Real-time image reconstruction from dense binary pixels," *arXiv e-print*, 2015. [Online]. Available: http://arxiv.org/abs/1510.04601

[10] W. Feng and Y. Chen, "Fast and accurate Poisson denoising with optimized nonlinear diffusion," *arXiv e-print*, 2015. [Online]. Available: http://arxiv.org/abs/1510.02930

[11] B. Zhang, J. M. Fadili, and J.-L. Starck, "Wavelets, ridgelets, and curvelets for Poisson noise removal," *IEEE Transaction on Image Processing*, vol. 17, no. 7, pp. 1093–1108, 2008. [Online]. Available: http://dx.doi.org/10.1109/TIP.2008.924386

[12] S. K. Bar-Lev and P. Enis, "On the construction of classes of variance stabilizing transformations," *Statistics & Probability Letters*, vol. 10, no. 2, pp. 95–100, 1990. [Online]. Available: http://dx.doi.org/10.1016/0167-7152(90)90002-O

[13] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "BM3D image denoising with shape-adaptive principal component analysis," in *Proc. SPARS'09 - Signal Processing with Adaptive Sparse Structured Representations*, Saint Malo, France, 2009, [Online]. Available: http://hal.inria.fr/inria-00369582. Matlab software at http://www.cs.tut.fi/~foi/GCF-BM3D/.

[14] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images," *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1395–1411, 2007, [Online]. Available: http://doi.org/10.1109/TIP.2007.891788. Matlab software at http://www.cs.tut.fi/~foi/SA-DCT/.

[15] A. Buades, B. Coll, and J. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling and Simulation*, vol. 4, no. 2, pp. 490–530, 2005, [Online]. Available: http://dx.doi.org/10.1137/040616024. Matlab software at http://www.mathworks.com/matlabcentral/fileexchange/13176-non-local-means-filter.

[16] A. Foi and G. Boracchi, "Foveated nonlocal self-similarity," *International Journal of Computer Vision*, pp. 1–33, 2016, in press. [Online]. Available: http://dx.doi.org/10.1007/s11263-016-0898-1. Matlab software at http://www.cs.tut.fi/~foi/FoveatedNL/.

[17] J. Boulanger, J.-B. Sibarita, C. Kervrann, and P. Bouthemy, "Non-parametric regression for patch-based fluorescence microscopy image sequence denoising," in *5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI 2008)*, 2008, pp. 748–751, [Online]. Available: http://dx.doi.org/10.1109/ISBI.2008.4541104. Software at http://serpico.rennes.inria.fr/doku.php?id=software:nd-safir:index.

[18] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Transactions on Image Processing*, vol. 12, no. 11, pp. 1338–1351, Nov 2003, [Online]. Available: http://dx.doi.org/10.1109/TIP.2003.818640. Matlab software at http://www.io.csic.es/PagsPers/JPortilla/software/file/3-bls-gsm-image-denoising-toolbox-in-matlab.

[19] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006, [Online]. Available: http://dx.doi.org/10.1109/TSP.2006.881199. Matlab software at http://www.cs.technion.ac.il/~elad/software.

[20] I. Ram, M. Elad, and I. Cohen, "Image denoising using NL-means via smooth patch ordering," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 1350–1354, [Online]. Available: http://dx.doi.org/10.1109/ICASSP.2013.6637871. Matlab software at http://www.cs.technion.ac.il/~elad/software.

[21] L. Azzari and A. Foi, Supplementary to the manuscript "Variance stabilization for noisy+estimate combination in iterative Poisson denoising", 2016. [Online]. Available: http://www.cs.tut.fi/~foi/invansc/

[22] Y. Chen, S. Fomel, and J. Hu, "Iterative deblending of simultaneous-source seismic data using seislet-domain shaping regularization," *Geophysics*, vol. 79, no. 5, pp. V179–V189, 2014. [Online]. Available: http://reproducibility.org/RSF/book/tccs/deblend/paper.pdf

[23] Y. Chen, L. Zhang, and L.-w. Mo, "Seismic data interpolation using nonlinear shaping regularization," *Journal of Seismic Exploration*, 2015. [Online]. Available: http://reproducibility.org/RSF/book/tccs/intshape/paper.pdf

[24] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error measurement to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004. [Online]. Available: http://dx.doi.org/10.1109/TIP.2003.819861

[25] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011. [Online]. Available: http://dx.doi.org/10.1561/2200000016

[26] J.-L. Starck, F. Murtagh, and A. Bijaoui, *Image Processing and Data Analysis: The Multiscale Approach*. Cambridge University Press, 1998, Appendix A. [Online]. Available: http://www.multiresolution.com/cupbook.pdf