

# Poisson Inverse Problems by the Plug-and-Play scheme

Arie Rond, Raja Giryes and Michael Elad

Department of Computer Science Technion—Israel Institute of Technology

Technion City, Haifa 32000, Israel

sarikr@cs.technion.ac.il | raja.giryes@duke.edu | elad@cs.technion.ac.il

November 10, 2015

## Abstract

The Anscombe transform [1] offers an approximate conversion of a Poisson random variable into a Gaussian one. This transform is important and appealing, as it is easy to compute, and becomes handy in various inverse problems with Poisson noise contamination. Solution to such problems can be done by first applying the Anscombe transform, then applying a Gaussian-noise-oriented restoration algorithm of choice, and finally applying an inverse Anscombe transform. The appeal in this approach is due to the abundance of high-performance restoration algorithms designed for white additive Gaussian noise (we will refer to these hereafter as "Gaussian-solvers"). This process is known to work well for high SNR images, where the Anscombe transform provides a rather accurate approximation. When the noise level is high, the above path loses much of its effectiveness, and the common practice is to replace it with a direct treatment of the Poisson distribution. Naturally, with this we lose the ability to leverage on vastly available Gaussian-solvers.

In this work we suggest a novel method for coupling Gaussian denoising algorithms to Poisson noisy inverse problems, which is based on a general approach termed "Plug-and-Play" [18]. Deploying the Plug-and-Play approach to such problems leads to an iterative scheme that repeats several key steps: (i) A convex programming task of simple form that can be easily treated; (ii) A powerful Gaussian denoising algorithm of choice; and (iii) A simple update step. Such a modular method, just like the Anscombe transform, enables other developers to plug their own Gaussian denoising algorithms to our scheme in an easy way. While the proposed method bares some similarity to the Anscombe operation, it is in fact based on a different mathematical basis, which holds true for all SNR ranges.

## 1 Introduction

In an inverse problem we are given a degraded image,  $y$ , and want to recover from it a clean image,  $x$ . The mathematical relation between the two images is given by  $y = \mathcal{N}(Hx)$ , where  $H$  is some linear operator and  $\mathcal{N}$  is a noise operator. A popular way to handle this reconstruction is to use a Bayesian probabilistic model that contains two ingredients: (i) the measurement forward model, mathematically given by  $P(y|x)$ ; and (ii) a prior model for clean images, given by  $P(x)$ .

In our work we concentrate on the case of Poisson Inverse Problems (PIP), Where  $\mathcal{N}$  stands for Poisson contamination. In a Poisson model for an image the gray levels of the image pixels are viewed as Poisson distributed random variables. More specifically, given a clean image pixel  $x[i]$ , the probability of getting a noisy value  $y[i]$  is given by

$$P(y[i] | x[i]) = \begin{cases} \frac{(x[i])^{y[i]}}{y[i]!} e^{-x[i]} & \text{if } x[i] > 0 \\ \delta(y[i]) & \text{if } x[i] = 0 \end{cases}. \quad (1)$$

A known property of this distribution is that  $x[i]$  is both the mean and variance of  $y[i]$ . This model is relevant in various tasks such as very low light imaging, CT reconstruction [13], fluorescence microscopy [2], astrophysics [16] and spectral imaging [10]. Common to all these tasks is the weak measured signal intensity.

An important note about Poisson noise is that the SNR of the measurements is proportional to the original image intensity, given by  $\sqrt{x[i]}$ . Therefore the peak value of an image is an important characteristic, needed when evaluating the level of noise in the image. For high peak levels, there exist several very effective ways to solve Poisson inverse problems. Many of these methods rely on the fact that it is possible to perform an approximate transform (known as Variance Stabilized Transform - VST) of the Poisson distribution into a Gaussian one [1], [8]. Since there are highly

\*This research was supported by the European Research Council under EUs 7th Framework Program, ERC grant agreement 320649, by the Intel Collaborative Research Institute for Computational Intelligence, and by the Google Faculty Research Award.

effective algorithms for Gaussian noise restoration (e.g. [4], [7], [11], [19], [5]), such methods can be used, followed by an inversion of the VST operation after the Gaussian solver [12], [20].

When dealing with lower peaks, such transformations become less efficient, and alternative methods are required, which treat the Poisson noise directly (e.g. [9], [15]). In recent years this direct approach has drawn considerable attention, and it seems to be very successful. In this work we aim at studying yet another method for Poisson inverse problem restoration that belongs to the direct approach family. The appeal of the proposed method is the fact that it offers an elegant bridge between the two families of methods, as it is relying too on Gaussian noise removal, applied iteratively.

This paper is organized in the following way : In section 2 we introduce the plug and play approach, as presented in [18]. We also extend this scheme to be able to work with several priors in parallel. In section 3 we present our algorithm, as derived from the plug and play approach. This section explains how to integrate a custom Gaussian denoising algorithm of choice, and discusses several improvements that were added to the algorithm. In section 4 we present experiments results, and in section 5 we conclude our paper by suggesting further improvements.

## 2 The Plug-and-Play (PaP) Approach

### 2.1 Standard Plug-and-Play

The Plug-and-Play framework, proposed by Venkatakrishnan, Bouman and Wohlberg [18], allows simple integration between inversion problems and priors, by applying a Gaussian denoising algorithm, which corresponds to the used prior. One of the prime benefits in the PaP scheme is the fact that the prior to be used does not have to be explicitly formulated as a penalty expression. Instead, the idea is to split the prior from the inverse problem, a task that is done elegantly by the ADMM optimization method [3], and then the prior is deployed indirectly by activating a Gaussian denoising algorithm of choice.

The goal of the PaP framework is to maximize the posterior probability in an attempt to implement the MAP estimator. Mathematically, this translate to the following:

$$\max_{x \in R^{m \times n}} P(x|y) = \max_{x \in R^{m \times n}} \frac{P(y|x) P(x)}{P(y)} = \max_{x \in R^{m \times n}} P(y|x) P(x). \quad (2)$$

The above suggests to maximize the posterior probability  $P(x|y)$  with respect to the ideal image  $x$ , which is of size  $n \times m$  pixels. Taking element wise  $-\ln(\cdot)$  of this expression gives an equivalent problem of the form

$$\min_{x \in R^{m \times n}} -\ln(P(x|y)) = \min_{x \in R^{m \times n}} -\ln(P(y|x)) - \ln(P(x)). \quad (3)$$

In order to be consistent with [18] we denote  $l(x) = -\ln(P(y|x))$  and  $s(x) = -\ln(P(x))$ . Thus our task is to find  $x$  that solves the problem

$$\hat{x} = \arg \min_{x \in R^{m \times n}} l(x) + \beta s(x). \quad (4)$$

Note that  $y$  is constant in this minimization. Also, a parameter  $\beta$  was added to achieve more flexibility. By adding a variable splitting technique to the optimization problem we get

$$\begin{aligned} \hat{x} &= \arg \min_{x, v \in R^{m \times n}} l(x) + \beta s(v). \\ \text{s.t. } & x = v \end{aligned} \quad (5)$$

This problem can be solved using ADMM [3] by constructing an augmented Lagrangian which is given by

$$L_\lambda = l(x) + \beta s(v) + \frac{\lambda}{2} \|x - v + u\|_2^2 - \frac{\lambda}{2} \|u\|_2^2. \quad (6)$$

ADMM theory [3] states that minimizing (5) is equivalent to iterating until convergence over the following three steps:

$$\begin{aligned} x^{k+1} &= \arg \min_x L_\lambda(x, v^k, u^k), \\ v^{k+1} &= \arg \min_v L_\lambda(x^{k+1}, v, u^k), \\ u^{k+1} &= u^k + (x^{k+1} - v^{k+1}). \end{aligned} \quad (7)$$

By plugging  $L_\lambda$  we get

$$\begin{aligned} x^{k+1} &= \arg \min_x l(y|x) + \frac{\lambda}{2} \|x - (v^k - u^k)\|_2^2, \\ v^{k+1} &= \arg \min_v \frac{\lambda}{2} \|x^{k+1} + u^k - v\|_2^2 + \beta s(v), \\ u^{k+1} &= u^k + (x^{k+1} - v^{k+1}). \end{aligned} \quad (8)$$

The second step means applying a Gaussian denoising algorithm which assumes a prior  $s(v)$  on the image  $x^{k+1} + u^k$  with variance of  $\sigma^2 = \frac{\beta}{\lambda}$ . Therefore, as already mentioned above, we do not have to know the formulation of the prior explicitly, as we can simply use a Gaussian denoising algorithm which corresponds to it.

The first step is dependent on the inversion problem we are trying to solve. In the next section we show how Poisson inverse problems are connected to this step. We will see that in this case step 1 is convex and becomes easy to compute. When handling the Poisson Denoising problem, this steps becomes even simpler because it is also separable, thus leading to a scalar formula that resembles the Anscombe transform.

## 2.2 Plug-and-Play Extension

We now show a simple extension of the Plug-and-Play method that enables to use several Gaussian denoisers. We start from the following ADMM formulation, that follows Equation (5)

$$\begin{aligned} \arg \min & l(x) + \beta_1 s_1(v_1) + \beta_2 s_2(v_2), \\ \text{s.t. } & x = v_1, x = v_2 \end{aligned} \quad (9)$$

where  $s_1$  and  $s_2$  are two priors that we aim to use, and  $v_1$  and  $v_2$  are two auxiliary variables that will help in simplifying the solution of this problem. Following the steps taken above in the derivation of the PaP, we get

Step 1:

$$x^{k+1} = \arg \min_x l(x) + \lambda \left\| x - v_1^k + u_1^k \right\|_2^2 + \lambda \left\| x - v_2^k + u_2^k \right\|_2^2. \quad (10)$$

As in the original Plug-and-Play, this expression too is convex if  $l(x)$  is convex, and also separable if  $l(x)$  is separable.

Step 2:

$$\begin{aligned} v_1^k &= \arg \min_v \beta_1 s_1(v_1) + \lambda \left\| x^{k+1} - v_1 + u_1^k \right\|_2^2 \\ v_2^k &= \arg \min_v \beta_2 s_2(v_2) + \lambda \left\| x^{k+1} - v_2 + u_2^k \right\|_2^2 \end{aligned} \quad (11)$$

which are two Gaussian denoising steps, each using a different prior.

Step 3:

$$\begin{aligned} u_1^{k+1} &= u_1^k + x^{k+1} - v_1^{k+1} \\ u_2^{k+1} &= u_2^k + x^{k+1} - v_2^{k+1} \end{aligned} \quad (12)$$

Obviously, this scheme can be generalized to use as many priors as needed. The core idea behind this generalization is that often times we may encounter different priors that address different features of the unknown image, such as self-similarity, local smoothness or other structure, scale-invariance, and more. By merging two such priors into the PaP scheme, we may get an overall benefit, as they complement each other.

## 3 P<sup>4</sup>IP Algorithm

We now turn to introduce the "Plug-and-Play Prior for Poisson Inverse Problem" algorithm, P<sup>4</sup>IP in short, and how it uses the plug and play framework. We start by invoking the proper log-likelihood function  $l(x)$  into the above-described formulation, this way enabling the integration of Gaussian denoising algorithms to the Poisson inverse problems. Then we discuss two applications of our algorithm – the denoising and the deblurring scenarios.

### 3.1 The Proposed Algorithm

We denote an original (clean) image, with dimensions  $m \times n$ , by an  $m \times n$  column-stacked vector  $x$ . Similarly, we denote a noisy image by  $y$ . The  $i$ 's pixel in  $x$  (and respectively  $y$ ) is given by  $x[i]$  (respectively  $y[i]$ ). We also denote by  $H$  the linear degradation operator that is applied on the image, which could be a blur operator, down-scaling or even a tomographic projection. In order to proceed we should find an expression for  $l(x)$ . As mentioned before, this is given by taking  $-\ln(\cdot)$  of  $P(y|x)$ . When taking  $H$  into account we get

$$P(y|x) = \prod_i \frac{(Hx)_i^{y_i}}{\Gamma(y_i + 1)} e^{-(Hx)_i}. \quad (13)$$

Thus,  $l(x)$  is given by

$$l(x) = \ln(P(y|x)) = \sum_i \ln \left( \frac{(Hx)_i^{y_i}}{\Gamma(y_i + 1)} e^{-(Hx)_i} \right) = -y^T \ln(Hx) + 1^T Hx + \text{constant}. \quad (14)$$

Relying on equation (8), the first ADMM step in matrix form is therefore

$$\arg \min_x L_\lambda = \arg \min_x -y^T \ln(Hx) + 1^T Hx + \frac{\lambda}{2} \|x - v + u\|_2^2. \quad (15)$$

This expression is convex and can be solved quite efficiently by modern optimization methods. The final algorithm is shown in Algorithm 1.

---

#### Algorithm 1 - P<sup>4</sup>IP

---

**Input:** Distorted image  $y$ , Gaussian\_denoise( $\cdot$ ) function

Initialization: set  $k = 0, u^0 = 0, v^0 = \text{some initialization}$ ;

**while** !stopping criteria **do**

$$x^{k+1} = \arg \min_x -y^T \ln(Hx) + 1^T Hx + \frac{\lambda}{2} \|x - v^k + u^k\|_2^2$$

$$v^{k+1} = \text{Gaussian\_denoise}(x^{k+1} + u^k) \text{ with } \sigma^2 = \frac{\beta}{\lambda}$$

$$u^{k+1} = u^k + (x^{k+1} - v^{k+1})$$

$$k = k + 1$$

**end while**

**Output:** Reconstructed image  $x^k$

---

Obviously we could use the Plug-and-Play extension that employs several denoising methods, as shown in the previous section. Such a change requires only slight modifications to Algorithm 1.

#### 3.1.1 Poisson Denoising

For the special case of Poisson denoising  $H = I$ . In this case the first ADMM step is separable, which means that it could be solved for each pixel individually. Moreover, this step can be solved by the closed form solution

$$x^{k+1}[i] = \frac{(\lambda(v^k[i] - u^k[i]) - 1) + \sqrt{(\lambda(v^k[i] - u^k[i]) - 1)^2 + 4\lambda y[i]}}{2\lambda}, \quad (16)$$

where  $x^k[i]$  is the  $i$ 'th pixel of  $x^k$  (and  $v^k[i]$ ,  $u^k[i]$  and  $y[i]$  are the  $i$ 'th pixels of  $v^k$ ,  $u^k$  and  $y$  respectively). The full derivation of this step is shown in the appendix A. A closer look at this expression reveals some resemblance to the Anscombe transform. Indeed, for the initial condition  $u^0 = 0, v^0 = 4\left(\sqrt{\frac{3}{8}} + 1\right)$ , and  $\lambda = 0.25$ , the variance of  $x$  is the same as the one achieved by Anscombe's transform, because they differ only by a constant. We mention here that we did not find that the initialization of the parameters lead to noticeable change in the final reconstruction, as long as it is the same order of magnitude of the noisy image, and therefore, all the shown results use all zero image as initialization. Figure 1 shows the transform done by Equation (16) for  $\lambda = 0.25, v^k[i] - u^k[i] = 4\left(\sqrt{\frac{3}{8}} + 1\right) + i$  for  $i = \{0, 3, 6, 9\}$ , and the Anscombe transform. While this curve may look like the Anscombe one, PaP is substantially different in two ways - (i) this curve changes (locally) from one iteration to another due to the change in  $u$  and  $v$ , and (ii) we do not apply the inverse transform after the Gaussian denoising.

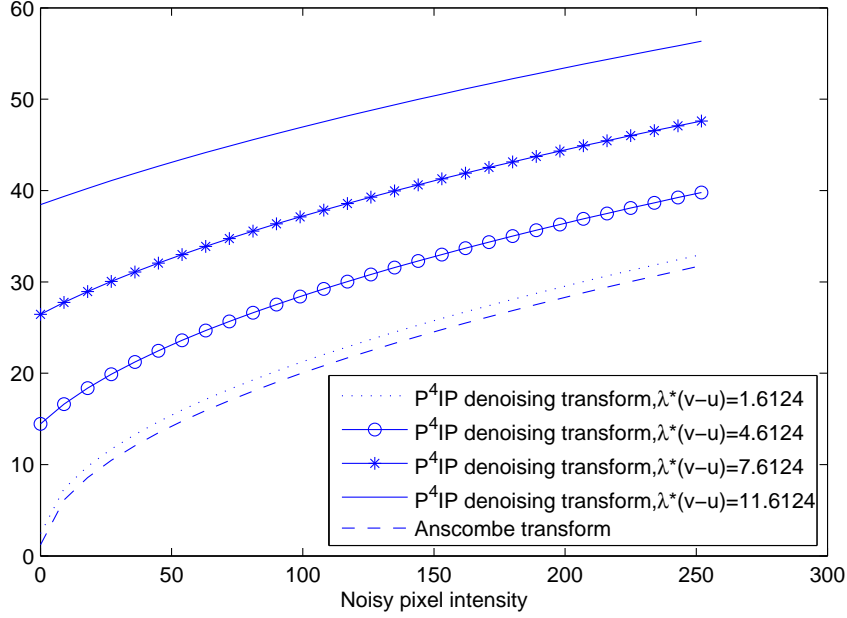


Figure 1:  $P^4IP$  and Anscombe transformations as a function of input noisy pixel.  $\lambda = 0.25$ .

### 3.1.2 Poisson Deblurring

When dealing with the deblurring problem,  $H$  represents a blur matrix. The first ADMM step is no longer separable and usually no analytical solution is available. However the problem is convex and a common way to solve it is by using iterative optimization methods, which usually require the gradient. Turning back to our problem, the gradient of  $L_\lambda(x)$  is given by

$$\nabla_x L_\lambda = -H^T (y / (Hx)) + H^T 1 + \lambda (x - v + u). \quad (17)$$

Where “/” stands for element-wise division. As can be seen, this gradient is easy to compute, as it requires blurring the temporary solution  $x$ , the constant vector 1 and the vector  $y / (Hx)$  in each such computation.

### 3.2 Details and Improvements

We chose to focus on two different inverse problems - the denoising and deblurring problem scenarios. In both, we chose BM3D as our Gaussian denoiser, as it provides very good results and has an efficient implementation. Both scenarios required appropriate choice of parameters and their values. An important parameter is  $\beta$  - the prior weight. Empirical results show that inappropriate choice of  $\beta$  leads to poor results, sometimes by several dB. Another two parameters that has big effect on the reconstruction relate to the choice of  $\lambda$ . This parameter is considered to be proportional to the inverse of the step size. We found that increasing  $\lambda$  at each iteration gives better results then a constant  $\lambda$ . Thus,  $\lambda$  update requires two parameters. The first is  $\lambda_0$  - the initial value of  $\lambda$ , and the second,  $\lambda_{step}$  - the value  $\lambda$  is multiplied by at each iteration. Another important parameter is the number of iterations. We chose to use a fixed number of iterations, but of course this parameter can also be learned or even estimated, similarly to what is done in [14]. For a given noise peak, each parameter was tuned on a series of 8 images. Out of each original image we generated 5 degraded images, noisy for the denoising scenario, and blurred and noisy for the deblurring scenario. We tested multiple peak values in the range of 0.2 - 4. We found that the optimal  $\lambda_0$  and  $\beta$  have strong correlation to the peak value. On the other hand,  $\lambda_{step}$  has a weak dependence on the peak, and was thus chosen independently of it.

Another improvement used, called binning, gives significant improvement in very low SNR cases. Here the image is down sampled and the algorithm is applied on the smaller sized image that has a better SNR, since we add up the photon counts of the merged pixels. Once the final result of the algorithm is obtained, we apply up-scaling by a simple linear interpolation. This technique leads to better results, and also reduces runtime as we are operating on smaller images. All the experiments reported below with binning use a 3:1 shown-scaling in each axis. Binning can only be used in the denoising scenario as down sampling doesn’t commute with the blur operator.

In the denoising scenario we also tested the multiple prior  $P^4IP$  Algorithm. We call this variation M- $P^4IP$ . As our second denoiser we chose to use a simplified version of [17], which is a multi-scale denoising algorithm. Multi-scale considerations are not used in BM3D, and therefore the two algorithms joint together may form a more powerful denoising prior.

In the deblurring scenario, L-BFGS was chosen as our optimization method. In order to avoid calculating  $\ln(\cdot)$  where  $Hx$  is negative, we optimized the surrogate function

$$f(x) = \begin{cases} L_\lambda(x) & , x < \varepsilon \\ ax^2 + bx + c & , x \geq \varepsilon \end{cases} \quad (18)$$

where the coefficients  $a, b$  and  $c$  were chosen such that this function and its derivative coincides with  $L_\lambda$  and its derivative at  $x = \varepsilon$ . as  $x \rightarrow 0$  we get that  $Hx \rightarrow 0$  and  $L_\lambda(x) \rightarrow \inf$ , therefore choosing a small enough  $\varepsilon$  value, guarantees that the surrogate function will have the same minimum as  $L_\lambda$  and all entries in  $Hx$  are positive.

Another technique that improves the results in both denoising and deblurring states that we can apply several algorithm runs with slightly different parameters and average the final results. Of course, this comes at cost of run time. All results shown here are without this trick.

## 4 Experiments

### 4.1 Denoising

We tested our algorithm for peak values 0.1, 0.2, 0.5, 1, 2 and 4. To evaluate our algorithm we compared to BM3D with the refined inverse Anscombe transform [12]. We also compared to [9], which leads to the best of our knowledge, to state of the art results. All algorithms were tested with and without binning. The results are shown in Table 1.

Figures 2-5 show several such results. As can be seen, the propose approach competes favorably with the BM3D+Anscombe and state-of-the-art algorithms. Binning is found to be beneficial for all algorithms when dealing with low peak values. As for run-times, our algorithm takes on roughly 30 sec/image when binning is used. This should be compared to the BM3D+Anscombe that runs somewhat faster (0.5 sec/image), and the SPDA method [9] which is much slower (an average of 15-20 minutes/image). When removing the binning, our algorithm runs for few minutes, BM3D+Anscombe runs for several seconds and SPDA runs for approximately 10 hours. These runtime evaluations were measured on an i7 with 8GB RAM laptop.

As mentioned before, to check the effectivity M-P&P we chose to combine BM3D with a simplified version of [17]. We noticed that it is important to find the right prior weight parameter. We only tested on a peak=0.2 scenario and the results are shown in table 3. For the tested peak, the algorithm gained 0.2 dB improvement. We note that it was harder to find good parameters and therefore we believe that it is possible to improve even more.

**Table 1** denoising without binning PSNR values

Method	Peak	Saturn	Flag	Camera	House	Swoosh	Peppers	Bridge	Ridges		Average
BM3D	0.1	19.42	13.05	15.66	16.28	16.93	15.61	15.68	<b>20.06</b>		16.59
SPDA		17.40	<b>13.35</b>	14.36	14.84	15.12	14.28	14.60	19.86		15.48
P <sup>4</sup> IP		<b>21.55</b>	13.30	<b>16.88</b>	<b>18.30</b>	<b>20.93</b>	<b>16.28</b>	<b>16.45</b>	19.08		<b>17.85</b>
BM3D	0.2	22.02	14.28	17.35	18.37	19.95	17.10	17.09	21.27		18.43
SPDA		21.52	<b>16.58</b>	16.93	17.83	18.91	16.75	16.80	<b>23.25</b>		18.57
P <sup>4</sup> IP		<b>23.05</b>	14.82	<b>17.82</b>	<b>19.48</b>	<b>23.34</b>	<b>17.31</b>	<b>17.54</b>	21.28		<b>19.33</b>
BM3D	0.5	23.86	15.87	18.83	20.27	22.92	18.49	18.24	23.37		20.23
SPDA		<b>25.50</b>	<b>19.67</b>	18.90	20.51	24.21	18.66	18.46	<b>27.76</b>		<b>21.71</b>
P <sup>4</sup> IP		25.19	16.50	<b>19.27</b>	<b>20.93</b>	<b>25.58</b>	<b>18.86</b>	<b>18.47</b>	23.57		21.05
BM3D	1	25.89	18.31	20.37	22.35	26.07	19.89	19.22	26.26		21.73
SPDA		27.02	<b>22.54</b>	20.23	<b>22.73</b>	26.28	19.99	19.20	<b>30.93</b>		<b>23.61</b>
P <sup>4</sup> IP		<b>27.05</b>	19.07	<b>20.54</b>	22.67	<b>27.79</b>	<b>20.07</b>	<b>19.31</b>	26.56		22.88
BM3D	2	27.42	20.81	<b>22.13</b>	24.18	28.09	<b>21.97</b>	<b>20.31</b>	29.82		23.56
SPDA		<b>29.38</b>	<b>24.92</b>	21.54	<b>25.09</b>	29.27	21.23	20.15	<b>33.40</b>		<b>25.62</b>
P <sup>4</sup> IP		28.93	21.04	21.87	24.65	<b>29.65</b>	21.33	20.16	29.97		24.70
BM3D	4	29.40	23.04	<b>23.94</b>	26.04	30.72	<b>24.07</b>	<b>21.50</b>	32.39		26.39
SPDA		<b>31.04</b>	<b>26.27</b>	21.90	26.09	<b>33.20</b>	22.09	20.55	<b>36.05</b>		<b>27.15</b>
P <sup>4</sup> IP		30.82	22.49	23.29	<b>26.33</b>	31.80	23.88	21.11	31.98		26.46

**Table 2** denoising with binning for peak 0.2 PSNR values

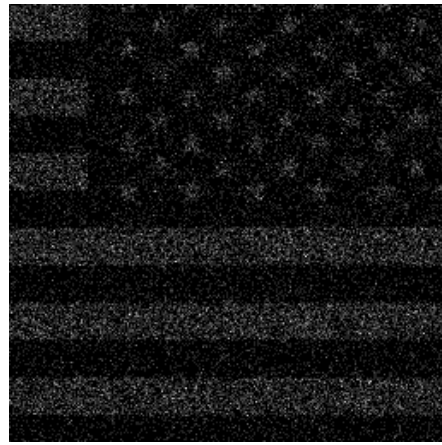
Method	Peak	Saturn	Flag	Camera	House	Swoosh	Peppers	Bridge	Ridges	Average
BM3Dbin	0.2	23.20	16.28	18.25	19.71	24.25	17.44	17.70	23.92	20.09
SPDAbin		<b>23.99</b>	<b>18.26</b>	17.95	19.62	23.53	<b>17.59</b>	<b>17.82</b>	<b>27.22</b>	<b>20.75</b>
P <sup>4</sup> IP bin		23.79	17.26	<b>18.58</b>	<b>19.96</b>	<b>24.53</b>	17.44	17.54	23.94	20.38

**Table 3** multiple priors PSNR values

Method	Peak	Saturn	Flag	Camera	House	Swoosh	Peppers	Bridge	Ridges	Average
P <sup>4</sup> IP bin	0.2	23.79	<b>17.26</b>	<b>18.58</b>	19.96	24.53	17.44	17.54	23.94	20.38
M-P <sup>4</sup> IP bin		<b>24.10</b>	16.77	<b>18.58</b>	<b>20.02</b>	<b>24.58</b>	<b>17.63</b>	<b>17.69</b>	<b>25.38</b>	<b>20.59</b>



original



noisy, peak=1



Anscome+BM3D, PSNR=18.51

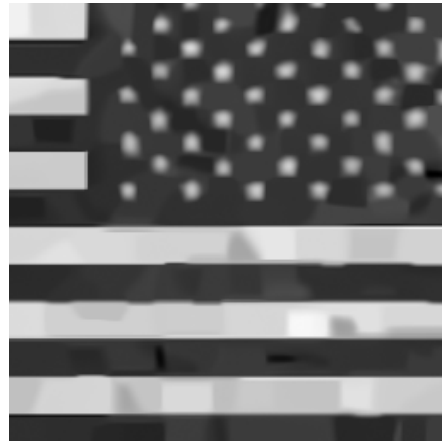
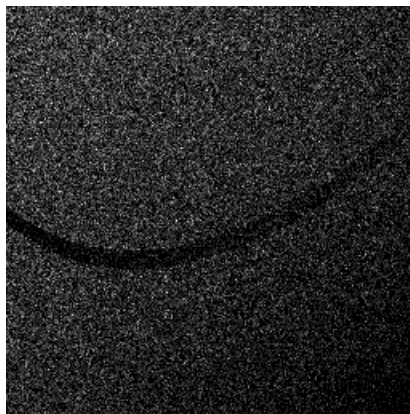
P<sup>4</sup>IP, PSNR=19.33

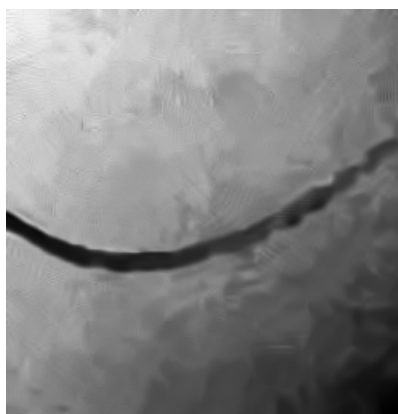
Figure 2: The image Flag with peak 1 - Denoising (no binning) results.



original



noisy, peak=2



Anscome+BM3D, PSNR=18.51



P<sup>4</sup>IP, PSNR=19.33

Figure 3: Peak 2 - Denoising (no binning) results





original

(a) noisy, peak=0.2



Anscome+BM3D, PSNR=19.90

(b) M-P<sup>4</sup>IP, PSNR=20.43

Figure 4: Peak 0.2 denoising (with binning)

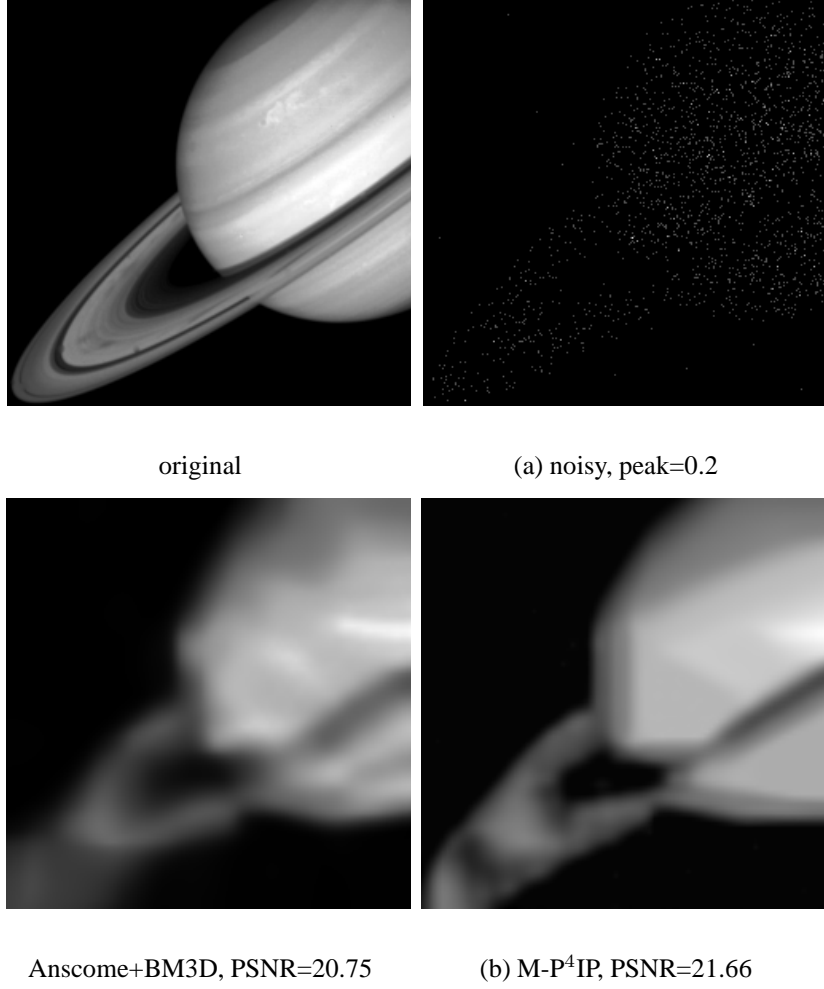


Figure 5: Peak 0.2 denoising (with binning)

## 4.2 Deblurring

In this scenario, we tested our algorithm for the peak values 1, 2 and 4 of an image that was blurred by one of the following blur kernels:

(i) a Gaussian kernel of size 25 by 25 with  $\sigma = 1.6$ .

(ii)  $\frac{1}{(1+x_1^2+x_2^2)}$  for  $x_1, x_2 = -7, \dots, 7$

(iii)  $9 \times 9$  uniform

To evaluate our algorithm we compared to IDD-BM3D [5] with the refined inverse Anscombe transform [12]. The results are shown in Tables 4, 5 and 6. Figures 6, 7 and 8 show specific results to better assess the visual quality of the outcome.

**Table 4** deblurring PSNR values for blur kernel (i)

Method	Peak	Saturn	Flag	Camera	House	Swoosh	Peppers	Bridge	Ridges		Average
BM3D	1	24.32	16.18	19.39	21.06	<b>26.51</b>	18.47	18.34	22.06		20.79
P <sup>4</sup> IP		<b>25.69</b>	<b>17.97</b>	<b>19.84</b>	<b>21.93</b>	<b>26.51</b>	<b>19.48</b>	<b>19.03</b>	<b>25.56</b>		<b>22.00</b>
BM3D	2	<b>26.07</b>	17.78	20.61	22.66	28.61	19.84	19.28	25.71		22.57
P <sup>4</sup> IP		25.95	<b>19.49</b>	<b>20.78</b>	<b>23.33</b>	<b>28.67</b>	<b>20.47</b>	<b>19.67</b>	<b>28.38</b>		<b>23.34</b>
BM3D	4	28.05	20.25	<b>21.66</b>	<b>24.69</b>	30.30	<b>21.25</b>	<b>20.20</b>	29.05		24.43
P <sup>4</sup> IP		<b>28.81</b>	<b>20.44</b>	21.37	24.51	<b>30.62</b>	21.11	20.13	<b>31.42</b>		<b>24.80</b>

**Table 5** deblurring PSNR values for blur kernel (ii)

Method	Peak	Saturn	Flag	Camera	House	Swoosh	Peppers	Bridge	Ridges		Average
BM3D	1	24.36	15.53	18.99	20.81	25.83	18.24	18.20	21.21		20.40
P <sup>4</sup> IP		<b>25.14</b>	<b>17.07</b>	<b>19.50</b>	<b>21.52</b>	<b>25.89</b>	<b>19.05</b>	<b>18.69</b>	<b>24.28</b>		<b>21.39</b>
BM3D	2	26.02	16.58	20.01	22.15	<b>28.33</b>	19.29	18.98	24.38		21.97
P <sup>4</sup> IP		<b>26.39</b>	<b>18.61</b>	<b>20.18</b>	<b>22.49</b>	28.29	<b>19.80</b>	<b>19.25</b>	<b>26.63</b>		<b>22.70</b>
BM3D	4	27.64	19.00	<b>20.84</b>	<b>23.68</b>	29.45	20.55	<b>19.71</b>	27.52		23.55
P <sup>4</sup> IP		<b>28.48</b>	<b>19.80</b>	20.76	23.58	<b>29.70</b>	<b>20.56</b>	19.70	<b>29.20</b>		<b>23.97</b>

**Table 6** deblurring PSNR values for blur kernel (iii)

Method	Peak	Saturn	Flag	Camera	House	Swoosh	Peppers	Bridge	Ridges		Average
BM3D	1	24.11	15.46	18.93	20.71	<b>26.23</b>	18.12	18.17	21.48		20.40
P <sup>4</sup> IP		<b>24.36</b>	<b>17.12</b>	<b>19.49</b>	<b>21.37</b>	26.03	<b>19.04</b>	<b>18.64</b>	<b>23.53</b>		<b>21.20</b>
BM3D	2	<b>26.06</b>	16.54	19.93	22.20	<b>28.26</b>	19.29	18.83	24.69		21.97
P <sup>4</sup> IP		25.62	<b>18.61</b>	<b>20.11</b>	<b>22.54</b>	28.17	<b>19.81</b>	<b>19.19</b>	<b>25.83</b>		<b>22.48</b>
BM3D	4	27.41	18.83	20.63	<b>23.47</b>	29.81	20.36	19.63	27.56		23.46
P <sup>4</sup> IP		<b>27.97</b>	<b>19.77</b>	<b>20.66</b>	23.39	<b>29.93</b>	<b>20.47</b>	<b>19.71</b>	<b>29.15</b>		<b>23.88</b>

It is clearly shown that in this scenario P<sup>4</sup>IP outperforms the Anscombe-transform framework. The runtime for a single image took about 5 minutes on an i7, 8G RAM laptop, about twice slower than Anscombe, and took 44 iterations.

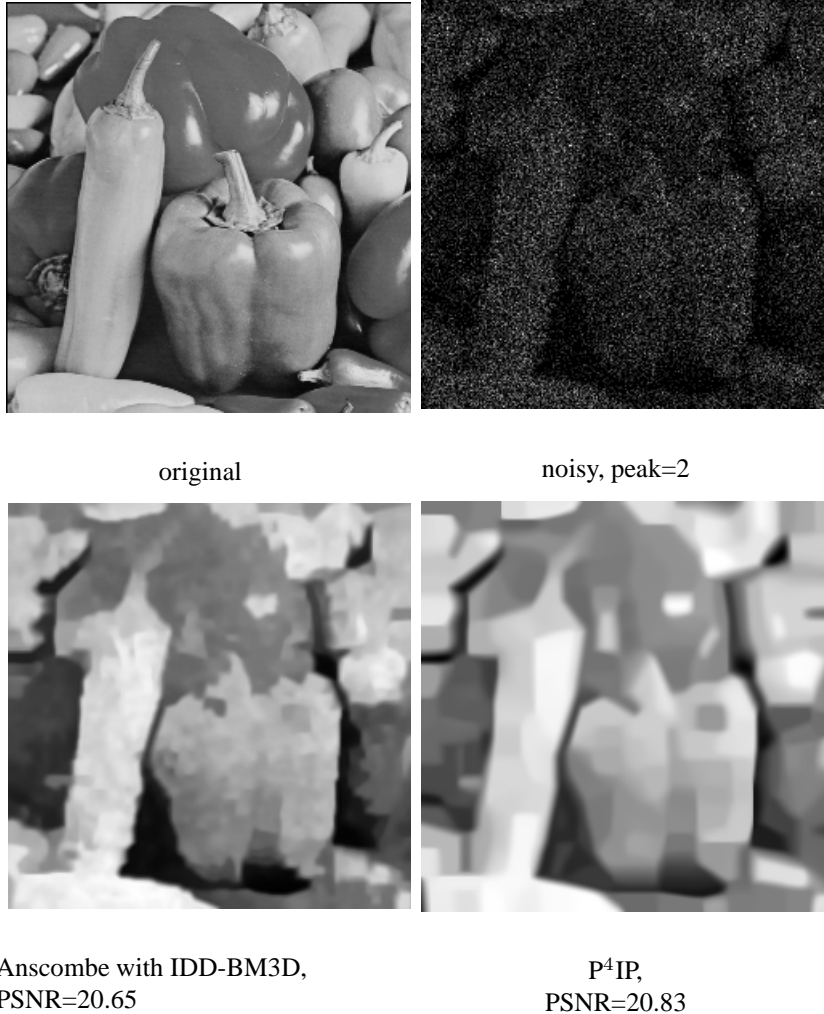
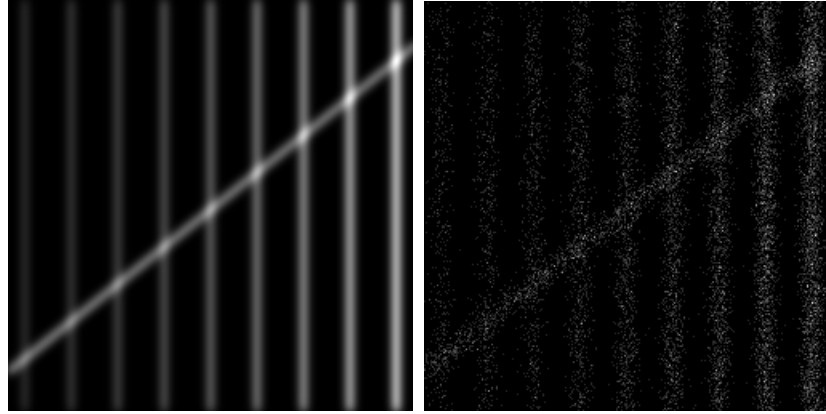
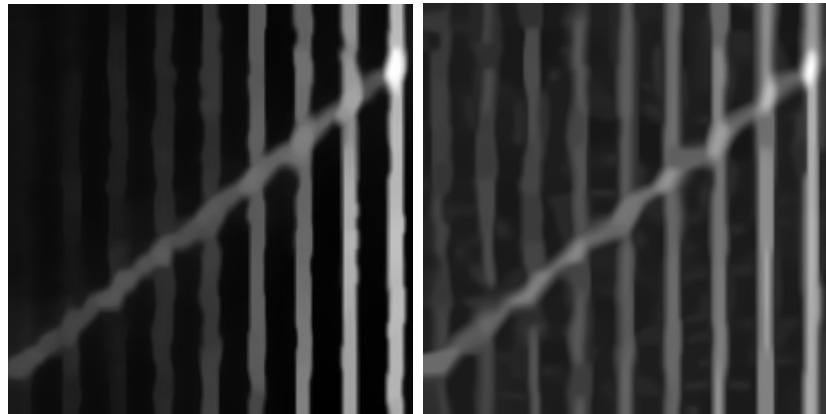


Figure 6: The image Peppers with peak 2 and blur kernel (i) - deblurring results.



original

noisy and blurry, peak=1



Anscombe with IDD-BM3D,  
PSNR=24.04

$P^4IP$ ,  
PSNR=26.56

Figure 7: The image Ridges with peak 2 and blur kernel (ii) - deblurring results.



Figure 8: The image Camera Man with peak 2 and blur kernel (iii) - deblurring results.

## 5 Conclusion and discussion

This work proposes a new way to integrate Gaussian denoising algorithms to Poisson noise inverse problems, by using the Plug-and-Play framework, this way taking advantage of the existing Gaussian solvers. The integration is done by simply using the Gaussian denoiser as a "black box" as part of the overall algorithm. This work demonstrates this paradigm on two problems - image denoising and image deblurring. Numerical results show that our algorithm outperforms the Anscombe-transform framework in lower peaks, and competes favorably with it on other cases. These results could be further improved by using the proposed extension of Plug-and-Play, which enables to combine multiple Gaussian denoising algorithms. Further work should be done in order to better tune the algorithm's parameters, similar to [6]. It is also interesting to learn more closely the relation between the Anscombe transform and our method. We have found that under certain initialization conditions, in the first step  $P^4IP$  does variance stabilization that is as good as Anscomb's one. It is possible that more could be said about the matter.

## Appendix A Derivation of first denoising ADMM step

In the denoising case  $H = I$  and we get that  $l(x)$  is given by

$$l(X) = -y^T \ln(x) + 1^T \ln(\Gamma(y+1)) + 1^T x. \quad (19)$$

The augmented Lagrangian is thus

$$L_\lambda = -y^T \ln(x) + 1^T x - \beta \ln(P(v)) + \frac{\lambda}{2} \|x - v + u\|^2 - \frac{\lambda}{2} \|u\|_2^2, \quad (20)$$

and the first ADMM step becomes

$$x^{k+1} = \arg \min_x L_\lambda(x, v^k, u^k) = \arg \min_x -y^T \ln(x) + 1^T x + \frac{\lambda}{2} \|x - v^k + u^k\|_2^2. \quad (21)$$

The first step ( $x$  update) is a convex and separable, implying that each entry of  $x$  can be treated separately. Furthermore, computing the elements of  $x$  is easily handled leading to a closed form expression. By differentiating  $L_\lambda$  by  $x[i]$  and equating to 0 we get

$$-\frac{y[i]}{x[i]} + 1 + \lambda(x[i] - v^k[i] + u^k[i]) = 0. \quad (22)$$

Thus, we get that

$$x[i] = \frac{(\lambda(v^k[i] - u^k[i]) - 1) + \sqrt{(\lambda(v^k[i] - u^k[i]) - 1)^2 + 4\lambda y[i]}}{2\lambda}. \quad (23)$$

As  $y$  is non negative, the expression inside the square root is also non negative and causes the resulted  $x$  to be non negative also. Another possible solution could have been the second root of Equation (22), but this solution is purely negative and thus uninformative.

## References

- [1] F. J. Anscombe. The transformation of poisson, binomial and negative-binomial data. *Biometrika*, pages 246–254, 1948.
- [2] J. Boulanger, C. Kervrann, P. Bouthemy, P. Elbau, J.-B. Sibarita, and J. Salamero. Patch-based nonlocal functional for denoising fluorescence microscopy image sequences. *Medical Imaging, IEEE Transactions on*, 29(2):442–454, 2010.
- [3] 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*, 3(1):1–122, 2011.
- [4] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16(8):2080–2095, 2007.
- [5] A. Danielyan, V. Katkovnik, and K. Egiazarian. Bm3d frames and variational image deblurring. *Image Processing, IEEE Transactions on*, 21(4):1715–1728, 2012.
- [6] C.-A. Deledalle, F. Tupin, and L. Denis. Poisson nl means: Unsupervised non local means for poisson noise. In *Image processing (ICIP), 2010 17th IEEE international conference on*, pages 801–804. IEEE, 2010.
- [7] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, 2006.
- [8] M. Fisz. The limiting distribution of a function of two independent random variables and its statistical application. In *Colloquium Mathematicae*, volume 3, pages 138–146. Institute of Mathematics Polish Academy of Sciences, 1955.
- [9] R. Giryes and M. Elad. Sparsity-based poisson denoising with dictionary learning. *Image Processing, IEEE Transactions on*, 23(12):5057–5069, 2014.
- [10] M. R. Keenan and P. G. Kotula. Accounting for poisson noise in the multivariate analysis of tof-sims spectrum images. *Surface and Interface Analysis*, 36(3):203–212, 2004.
- [11] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2272–2279. IEEE, 2009.
- [12] M. Makitalo and A. Foi. Optimal inversion of the anscombe transformation in low-count poisson image denoising. *Image Processing, IEEE Transactions on*, 20(1):99–109, 2011.
- [13] I. Rodrigues, J. Sanches, and J. Bioucas-Dias. Denoising of medical images corrupted by poisson noise. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 1756–1759. IEEE, 2008.

- [14] Y. Romano and M. Elad. Improving k-svd denoising by post-processing its method-noise. In *ICIP*, pages 435–439, 2013.
- [15] J. Salmon, Z. Harmany, C.-A. Deledalle, and R. Willett. Poisson noise reduction with non-local pca. *Journal of mathematical imaging and vision*, 48(2):279–294, 2014.
- [16] J. Schmitt, J. Starck, J. Casandjian, J. Fadili, and I. Grenier. Poisson denoising on the sphere: application to the fermi gamma ray space telescope. *Astronomy & Astrophysics*, 517:A26, 2010.
- [17] J. Sulam, B. Ophir, and M. Elad. Image denoising through multi-scale learnt dictionaries. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 808–812. IEEE, 2014.
- [18] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. Plug-and-play priors for model based reconstruction. 2013.
- [19] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: from gaussian mixture models to structured sparsity. *Image Processing, IEEE Transactions on*, 21(5):2481–2499, 2012.
- [20] B. Zhang, J. M. Fadili, and J.-L. Starck. Wavelets, ridgelets, and curvelets for poisson noise removal. *Image Processing, IEEE Transactions on*, 17(7):1093–1108, 2008.