Convergence of Plug-and-Play algorithms for image inverse problems

Samuel Hurault
ENS Paris

PhD supervisors: Arthur Leclaire, Nicolas Papadakis

Collaborations: Antonin Chambolle, Ulugbek Kamilov, Lingxiao Li, Justin Solomon, Matthieu Terris, Julian Tachella

Image Inverse Problems

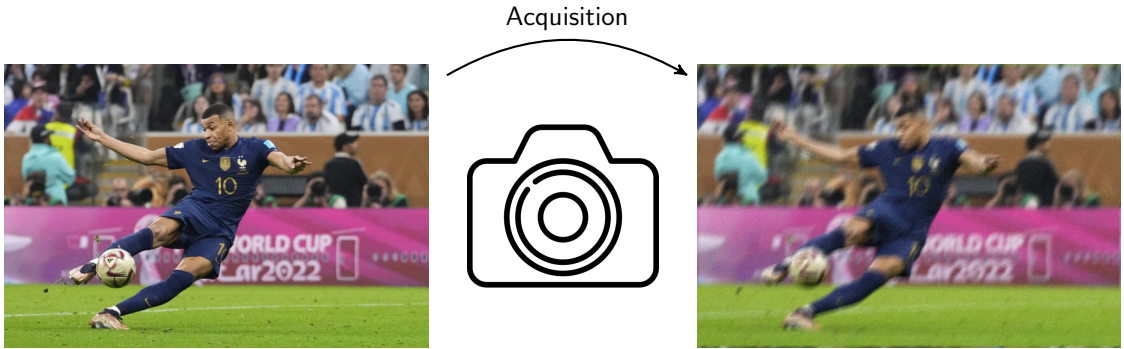


Image Inverse Problems

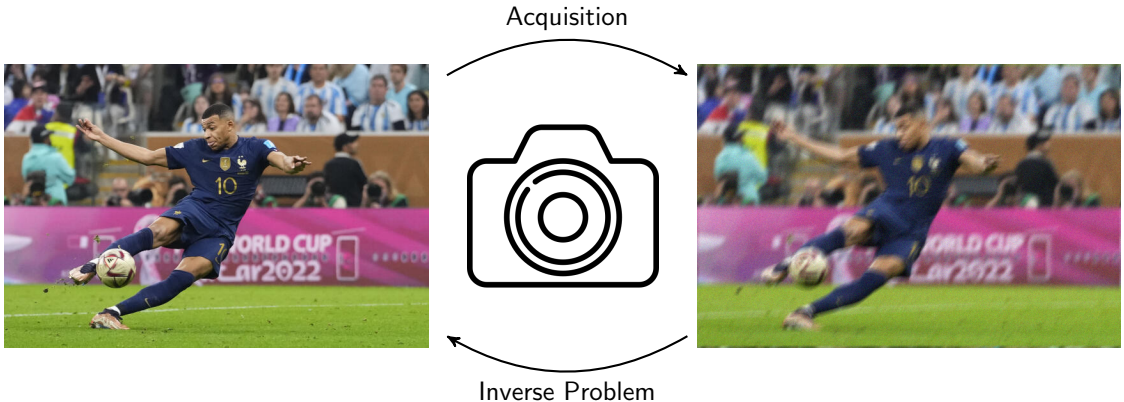
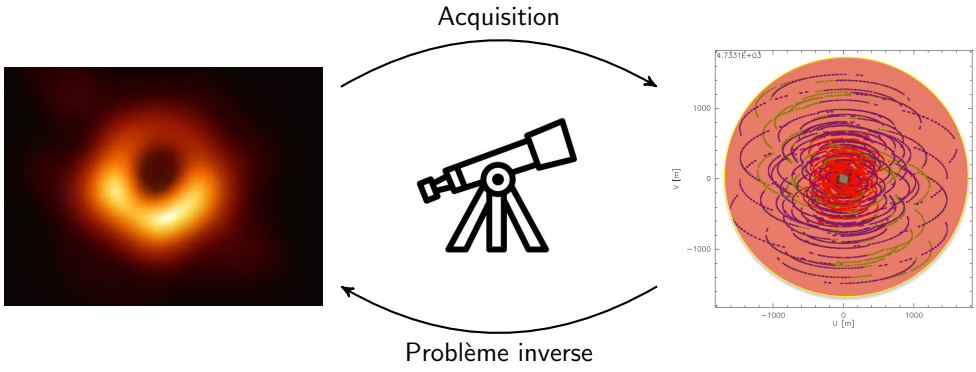


Image Inverse Problems

Degradation

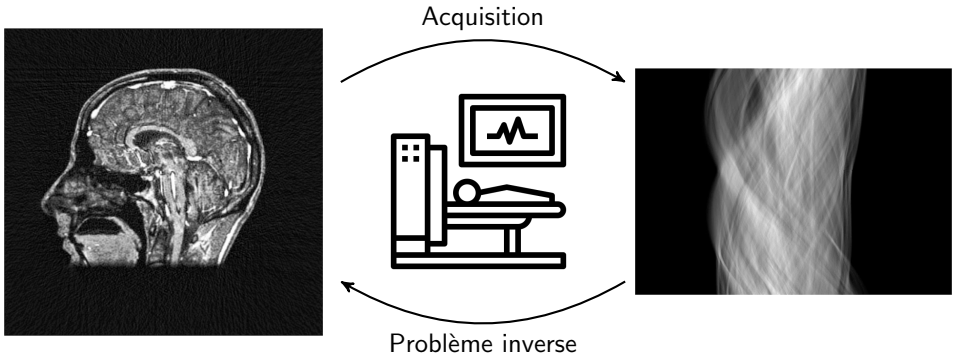
Inverse Problem

Image Inverse Problems



Sources: Wikipedia, [Isella, '11]

Image Inverse Problems

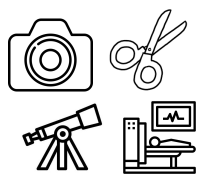


Source: Wikipedia

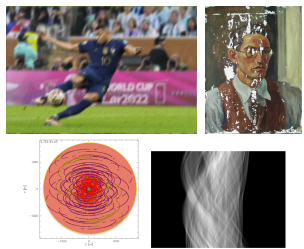
Image Inverse Problems

$x \in \mathbb{R}^n$

Acquisition / Degradation

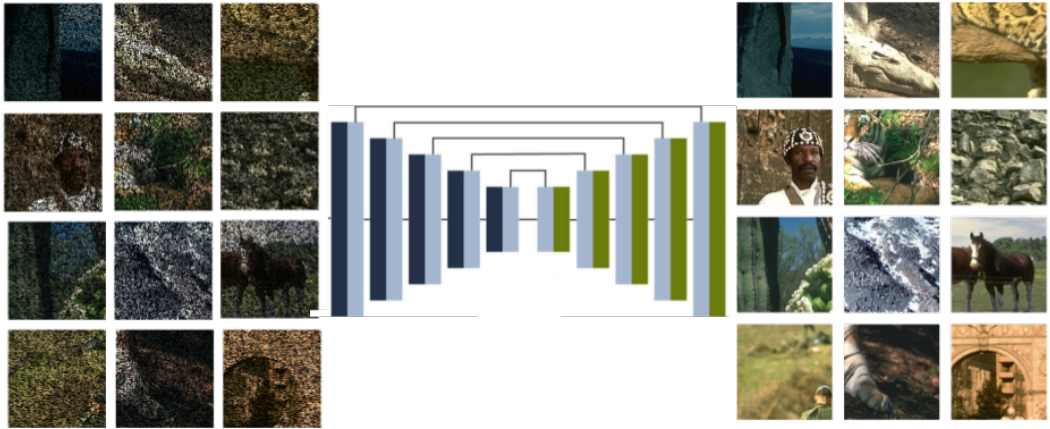


Inverse Problem



$y \sim \text{noise}(Ax) \in \mathbb{R}^m$

Supervised methods [Alder & Ökten, '18, Bertocchi et. al '18, Zhang et. al '20]



✓ If access to a dataset (x_i, y_i), state-of-the-art performance.

✗ Requires to retrain for each degradation A.

Variational formulation

Estimate x from observation $y \sim \text{noise}(Ax)$

Variational formulation

Estimate x from observation $y \sim \text{noise}(Ax)$

- Estimation x^* must verify:
  - Ax^* "is close from" y.
  - x^* "looks like" a real image.

Variational formulation

Estimate x from observation $y \sim \text{noise}(Ax)$

💡 Estimation x^* must verify:

- Ax^* "is close from" y.
- x^* "looks like" a real image.

$$x^* = \underset{x \in \mathbb{R}^n}{\arg\min} \quad f(x) \quad + \quad g(x)$$

Variational formulation

Estimate x from observation $y \sim \text{noise}(Ax)$

💡 Estimation x^* must verify:

- Ax^* "is close from" y.
- x^* "looks like" a real image.

$$x^* = \underset{x \in \mathbb{R}^n}{\arg\min} \quad f(x) \quad + \quad g(x)$$

Data-fidelity

ex: $f(x) = \frac{1}{2}||Ax - y||^2$

Variational formulation

Estimate x from observation $y \sim \text{noise}(Ax)$

💡 Estimation x^* must verify:

- Ax^* "is close from" y.
- x^* "looks like" a real image.

$$x^* = \arg\min_{x \in \mathbb{R}^n} \quad f(x) \quad + \quad g(x)$$

Data-fidelity

ex: $f(x) = \frac{1}{2}||Ax - y||^2$

Regularization

ex: Total Variation [Rudin et al. '92]
Wavelet sparcity [Mallat '09]
Deep prior [Bora et al. '17]

Variational formulation

Estimate x from observation $y \sim \text{noise}(Ax)$

Probabilistic formulation

Estimate x from observation $y \sim p(y|x)$

💡 Estimation x^* must verify:

- Ax^* "is close from" y.
- x^* "looks like" a real image.

$$x^* = \underset{x \in \mathbb{R}^n}{\arg\min} \quad f(x) \quad + \quad g(x)$$

Data-fidelity

ex: $f(x) = \frac{1}{2}||Ax - y||^2$

Regularization

ex: Total Variation [Rudin et al. '92]
Wavelet sparcity [Mallat '09]
Deep prior [Bora et al. '17]

Variational formulation

Estimate x from observation $y \sim \text{noise}(Ax)$

💡 Estimation x^* must verify:

- Ax^* "is close from" y.
- x^* "looks like" a real image.

$$x^* = \arg\min_{x \in \mathbb{R}^n} \quad f(x) \quad + \quad g(x)$$

Data-fidelity

ex: $f(x) = \frac{1}{2}||Ax - y||^2$

Regularization

ex: Total Variation [Rudin et al. '92]
Wavelet sparcity [Mallat '09]
Deep prior [Bora et al. '17]

Probabilistic formulation

Estimate x from observation $y \sim p(y|x)$

Maximum A-Posteriori

$$\begin{aligned} x^* &= \arg\max_{x \in \mathbb{R}^n} p(x|y) \\ &= \arg\min_{x \in \mathbb{R}^n} -\log p(y|x) \quad -\log p(x) \end{aligned}$$

Variational formulation

Estimate x from observation $y \sim \text{noise}(Ax)$

💡 Estimation x^* must verify:

- Ax^* "is close from" y.
- x^* "looks like" a real image.

$$x^* = \arg\min_{x \in \mathbb{R}^n} \quad f(x) \quad + \quad g(x)$$

Data-fidelity

ex: $f(x) = \frac{1}{2}||Ax - y||^2$

Regularization

ex: Total Variation [Rudin et al. '92]
Wavelet sparcity [Mallat '09]
Deep prior [Bora et al. '17]

Probabilistic formulation

Estimate x from observation $y \sim p(y|x)$

Maximum A-Posteriori

$$x^* = \arg\max_{x \in \mathbb{R}^n} p(x|y)$$

$$= \arg\min_{x \in \mathbb{R}^n} -\log p(y|x) \quad -\log p(x)$$

- Gaussian noise: $f(x) = \frac{1}{2}||Ax - y||^2$
- Poisson noise: $f(x) = \text{KL}(y, Ax)$

Variational formulation

Estimate x from observation $y \sim \text{noise}(Ax)$

Estimation x^* must verify:

- Ax^* "is close from" y.
- x^* "looks like" a real image.

$$x^* = \arg\min_{x \in \mathbb{R}^n} \quad f(x) \quad + \quad g(x)$$

Data-fidelity

ex: $f(x) = \frac{1}{2}||Ax - y||^2$

Regularization

ex: Total Variation [Rudin et al. '92]
Wavelet sparcity [Mallat '09]
Deep prior [Bora et al. '17]

Probabilistic formulation

Estimate x from observation $y \sim p(y|x)$

Maximum A-Posteriori

$$x^* = \arg\max_{x \in \mathbb{R}^n} p(x|y)$$

$$= \arg\min_{x \in \mathbb{R}^n} -\log p(y|x) \quad -\log p(x)$$

- Gaussian noise: $f(x) = \frac{1}{2}||Ax - y||^2$
- Poisson noise: $f(x) = \text{KL}(y, Ax)$

$g(x) = -\log p(x)$

Heuristics of Plug-and-Play methods [Venkatakrishnan et al. '13]

Estimate x from observation $y \sim \text{noise}(Ax)$

- **Decouple** optimization over f and g. [Combette & Pesquet '11] [Zoran & Weiss '11]
- Use a **Gaussian Denoiser** as the optimization step over the regularizer g.
  - → Easy Inverse problem $A = \text{Id}$, performant deep denoisers [Zhang et al. '17,'19,'21] [Song et al. '19]
  - → $g(\text{Denoiser(x)}) < g(x)$ **(Implicit Regularization)**

From $y \sim \text{noise}(Ax)$, iterate:

1. Denoising step
2. Decrease the data-fidelity term f.

✓ "One denoiser to solve them all!" [Chang et. al]

Presentation Outline

1. **Introduction to Plug-and-Play methods**
   - How can an image Denoiser be used to regularize an inverse problem?
2. **Convergence of Plug-and-Play algorithms**
   - Can sufficient conditions on the Denoiser be found to ensure that Plug-and-Play algorithms converge while maintaining performance?
3. **Bregman Generalization and inverse problems with Poisson noise**
   - How can these schemes be applied to less regular data fidelity terms?

Introduction to Plug-and-Play methods

First-order optimization

Find $\arg\min_{x \in \mathbb{R}^n} F(x)$

- Gradient Descent (explicit):

$$x_{k+1} = (\mathrm{Id} - \tau \nabla F)(x_k)$$

- Implicit Gradient Descent / Proximal Point Algorithm:

$$\begin{aligned} x_{k+1} &= x_k - \tau \nabla F(x_{k+1}) \\ \Leftrightarrow x_{k+1} &= \mathrm{Prox}_{\tau F}(x_k) \end{aligned}$$

where $\mathrm{Prox}_F(y) = \underset{x \in \mathbb{R}^n}{\arg\min} \dfrac{1}{2}||x-y||^2 + F(x)$

First-order optimization

Find $\arg\min_{x\in\mathbb{R}^n} F(x)$

- Gradient Descent (explicit):

$$x_{k+1} = (\mathrm{Id} - \tau\nabla F)(x_k)$$

- Implicit Gradient Descent / Proximal Point Algorithm:

$$\begin{aligned} x_{k+1} &= x_k - \tau\nabla F(x_{k+1}) \\ \Leftrightarrow x_{k+1} &= \mathrm{Prox}_{\tau F}(x_k) \end{aligned}$$

where $\mathrm{Prox}_F(y) = \underset{x\in\mathbb{R}^n}{\arg\min}\ \frac{1}{2}||x-y||^2 + F(x)$

Find $\arg\min_{x\in\mathbb{R}^n} f(x) + g(x)$

- Gradient descent:

$$x_{k+1} = (\mathrm{Id} - \tau\nabla(f+g))(x_k)$$

- Proximal Gradient Descent (PGD):

$$x_{k+1} = \mathrm{Prox}_{\tau g} \circ (\mathrm{Id} - \tau\nabla f)(x_k)$$

- Douglas-Rashford Splitting / ADMM:

$$x_{k+1} = \left(\frac{1}{2}\mathrm{Id} + \frac{1}{2}(2\mathrm{Prox}_{\tau g} - \mathrm{Id}) \circ (2\mathrm{Prox}_{\tau f} - \mathrm{Id})\right)$$

First-order optimization

Find $\arg\min_{x\in\mathbb{R}^n} F(x)$

- Gradient Descent (explicit):

$$x_{k+1} = (\mathrm{Id} - \tau\nabla F)(x_k)$$

- Implicit Gradient Descent / Proximal Point Algorithm:

$$\begin{aligned} x_{k+1} &= x_k - \tau\nabla F(x_{k+1}) \\ \Leftrightarrow x_{k+1} &= \mathrm{Prox}_{\tau F}(x_k) \end{aligned}$$

where $\mathrm{Prox}_F(y) = \underset{x\in\mathbb{R}^n}{\arg\min}\ \frac{1}{2}||x-y||^2 + F(x)$

Find $\arg\min_{x\in\mathbb{R}^n} f(x) + g(x)$

- Gradient descent:

$$x_{k+1} = (\mathrm{Id} - \tau\nabla(f+g))(x_k)$$

- Proximal Gradient Descent (PGD):

$$x_{k+1} = \mathrm{Prox}_{\tau g} \circ (\mathrm{Id} - \tau\nabla f)(x_k)$$

- Douglas-Rashford Splitting / ADMM:

$$x_{k+1} = \left(\frac{1}{2}\mathrm{Id} + \frac{1}{2}(2\mathrm{Prox}_{\tau g} - \mathrm{Id}) \circ (2\mathrm{Prox}_{\tau f} - \mathrm{Id})\right)$$

🤔 Involve ∇g or $\mathrm{Prox}_{\tau g}$ but not g explicitly.

Denoising: an implicit regularization

Estimate x from observation $y = x + \xi$

- $x \sim p(x)$.
- $\xi \sim \mathcal{N}(0, \sigma^2 \,\mathrm{Id})$ Gaussian noise .

Denoising: an implicit regularization

Estimate x from observation $y = x + \xi$

- $x \sim p(x)$.
- $\xi \sim \mathcal{N}(0, \sigma^2 \,\mathrm{Id})$ Gaussian noise .

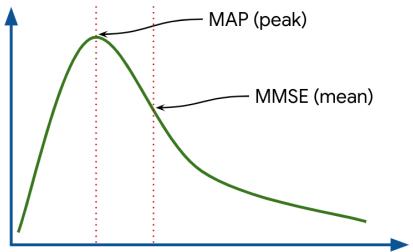
  $\rightarrow$ Two *theoretical* denoisers:

MMSE denoiser

$$D_\sigma^{\mathrm{MMSE}}(y) = \mathbb{E}_{x \sim p(x|y)}[x]$$

MAP Denoiser

$$D_\sigma^{\mathrm{MAP}}(y) = \arg\max_x p(x|y)$$



Denoising: an implicit regularization

Estimate x from observation $y = x + \xi$

- $x \sim p(x)$. $\quad p_\sigma = p * G_\sigma$ with G_σ Gaussian pdf of variance σ^2.
- $\xi \sim \mathcal{N}(0, \sigma^2 \,\mathrm{Id})$ Gaussian noise .

$\rightarrow$ Two *theoretical* denoisers:

MMSE denoiser

$$D_\sigma^{\mathrm{MMSE}}(y) = \mathbb{E}_{x\sim p(x|y)}[x]$$

MAP Denoiser

$$D_\sigma^{\mathrm{MAP}}(y) = \arg\max_x p(x|y)$$

Tweedie:

$$D_\sigma^{\mathrm{MMSE}} = \mathrm{Id} - \sigma^2 \nabla(-\log p_\sigma)$$

Bayes:

$$D_\sigma^{\mathrm{MAP}} = \mathrm{Prox}_{-\sigma^2 \log p}$$

Denoising: an implicit regularization

Estimate x from observation $y = x + \xi$

- $x \sim p(x)$. $\quad p_\sigma = p * G_\sigma$ with G_σ Gaussian pdf of variance σ^2.
- $\xi \sim \mathcal{N}(0, \sigma^2\,\mathrm{Id})$ Gaussian noise .

  $\rightarrow$ Two *theoretical* denoisers:

MMSE denoiser

$$D_\sigma^{\mathrm{MMSE}}(y) = \mathbb{E}_{x\sim p(x|y)}[x]$$

MAP Denoiser

$$D_\sigma^{\mathrm{MAP}}(y) = \arg\max_x p(x|y)$$

Tweedie:

$$D_\sigma^{\mathrm{MMSE}} = \mathrm{Id} - \sigma^2 \nabla(-\log p_\sigma)$$

Bayes:

$$D_\sigma^{\mathrm{MAP}} = \mathrm{Prox}_{-\sigma^2 \log p}$$

$\rightarrow$ Given a *real* denoiser D_σ:

$$D_\sigma \approx D_\sigma^{MMSE} \qquad \text{or} \qquad D_\sigma \approx D_\sigma^{MAP}$$

Denoising $\approx$ explicit or implicit descent on $g = -\log p$.

Small recap

1. $g = -\log p$ is the ideal regularization
2. Minimize $f + g$ requires access to ∇g or $\operatorname{Prox} g$.
3. A Gaussian denoiser approximates $\nabla(-\log p)$ or $\operatorname{Prox}_{-\log p}$.

Plug-and-play algorithms

Estimate $\arg\min_{x\in\mathbb{R}^n} f(x)+g(x)$ with $f=-\log p(y|.)$ and $g=-\log p$

$$\text{PGD:}\begin{cases} x_{k+1}=\text{Prox}_{\tau f}\circ(\text{Id}-\tau\nabla g)(x_k) \text{ or} \\ x_{k+1}=\text{Prox}_{\tau g}\circ(\text{Id}-\tau\nabla f)(x_k)\end{cases}$$

✗ unknown regularization g

Plug-and-play algorithms

Estimate $\arg\min_{x\in\mathbb{R}^n} f(x)+g(x)$ with $f=-\log p(y|.)$ and $g=-\log p$

$$\text{PGD:} \begin{cases} x_{k+1} = \text{Prox}_{\tau f}\circ(\text{Id}-\tau\nabla g)(x_k) \text{ or} \\ x_{k+1} = \text{Prox}_{\tau g}\circ(\text{Id}-\tau\nabla f)(x_k) \end{cases}$$

✗ unknown regularization g

MMSE Denoiser

$$D_\sigma^{MMSE} = \text{Id}-\sigma^2\nabla(-\log p_\sigma)$$

→ Given a *real* denoiser D_σ:

GradPnP algorithms (RED)

[Romano et al., '16]

$$\nabla g \leftarrow \text{Id}-D_\sigma$$

GradPnP-PGD:

$$x_{k+1} = \text{Prox}_{\tau f}\circ(\tau D_\sigma + (1-\tau)\,\text{Id})(x_k)$$

Plug-and-play algorithms

Estimate $\arg\min_{x\in\mathbb{R}^n} f(x)+g(x)$ with $f=-\log p(y|.)$ and $g=-\log p$

$$\text{PGD:}\begin{cases} x_{k+1}=\text{Prox}_{\tau f}\circ(\text{Id}-\tau\nabla g)(x_k) \text{ or} \\ x_{k+1}=\text{Prox}_{\tau g}\circ(\text{Id}-\tau\nabla f)(x_k)\end{cases}$$

✗ unknown regularization g

MMSE Denoiser

$$D_\sigma^{MMSE}=\text{Id}-\sigma^2\nabla(-\log p_\sigma)$$

MAP Denoiser

$$D_\sigma^{MAP}=\text{Prox}_{-\sigma^2\log p}$$

→ Given a *real* denoiser D_σ:

GradPnP algorithms (RED)

[Romano et al., '16]

$$\nabla g \leftarrow \text{Id}-D_\sigma$$

GradPnP-PGD:

$$x_{k+1}=\text{Prox}_{\tau f}\circ(\tau D_\sigma+(1-\tau)\,\text{Id})(x_k)$$

ProxPnP algorithms

[Venkatakrishnan et al., '13]

$$\text{Prox}_{\tau g}\leftarrow D_\sigma$$

ProxPnP-PGD:

$$x_{k+1}=D_\sigma\circ(\text{Id}-\tau\nabla f)(x_k)$$

Plug-and-play algorithms

Estimate x from observation $y \sim p(y|x)$

MAP / 1st order optim.

$$\arg\min_{x \in \mathbb{R}^n} f(x) + g(x)$$

✓ Converges.

✗ Unknown regularization g.

$$\xrightarrow[D_\sigma \approx \mathrm{Prox}_{\tau g}]{D_\sigma \approx \mathrm{Id} - \nabla g}$$

Plug-and-Play Algorithms

✓ Implicit regularization.

Plug-and-play algorithms

Estimate x from observation $y \sim p(y|x)$

MAP / 1st order optim.

$$\arg\min_{x \in \mathbb{R}^n} f(x) + g(x)$$

✓ Converges.

✗ Unknown regularization g.

$$\xrightarrow[D_\sigma \approx \mathrm{Prox}_{\tau g}]{D_\sigma \approx \mathrm{Id} - \nabla g}$$

Plug-and-Play Algorithms

✓ Implicit regularization.

✓ State-of-the-art performance.

DPIR [Zhang et al. '21]

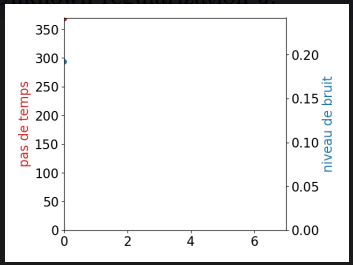
$D_\sigma \approx \mathrm{Id} - \nabla g$

$D_\sigma \approx \mathrm{Prox}_{\tau g}$

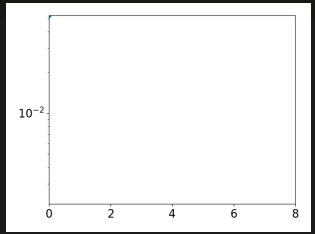
$\mathrm{PSNR}(x_k, x^*)$

✓

✗ Unknown regularization σ.



Parameters τ, σ



$$\frac{||x_{k+1} - x_k||}{||x_k||}$$

Plug-and-play algorithms

Estimate x from observation $y \sim p(y|x)$

MAP / 1st order optim.

$$\arg\min_{x \in \mathbb{R}^n} f(x) + g(x)$$

✓ Converges.

✗ Unknown regularization g.

$$\frac{D_\sigma \approx \mathrm{Id} - \nabla g}{D_\sigma \approx \mathrm{Prox}_{\tau g}} \longrightarrow$$

Plug-and-Play Algorithms

✓ Implicit regularization.

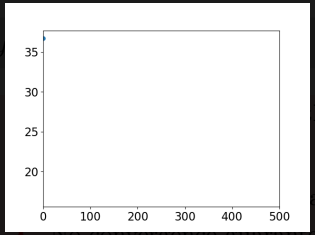
✓ State-of-the-art performance.

✗ No convergence guarantees.

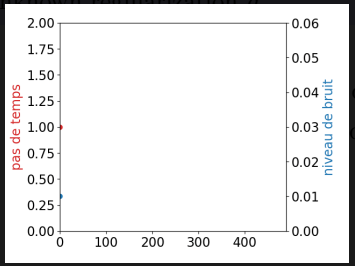
✗ For a generic denoiser (e.g.: a deep neural net):

$$D_\sigma \neq \mathrm{Id} - \nabla g \text{ and } D_\sigma \neq \mathrm{Prox}_{\tau g}.$$

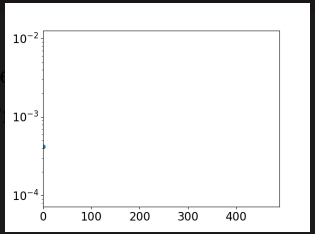
DPIR [Zhang et al. '21] + 500 iterations with fixed τ and σ.



PSNR(x_k, x^*)



Parameters τ, σ



$\frac{||x_{k+1}-x_k||}{||x_k||}$

Plug-and-play algorithms

Estimate x from observation $y \sim p(y|x)$

MAP / 1st order optim.

$$\arg\min_{x \in \mathbb{R}^n} f(x) + g(x)$$

✓ Converges.

✗ Unknown regularization g.

$$\xrightarrow[D_\sigma \approx \mathrm{Prox}_{\tau g}]{D_\sigma \approx \mathrm{Id} - \nabla g}$$

Plug-and-Play Algorithms

✓ Implicit regularization.

✓ State-of-the-art performance.

✗ No convergence guarantees.

✗ No minimized functional.

✗ For a generic denoiser (e.g.: a deep neural net):

$$D_\sigma \neq \mathrm{Id} - \nabla g \text{ and } D_\sigma \neq \mathrm{Prox}_{\tau g}.$$

Plug-and-play algorithms

Estimate x from observation $y \sim p(y|x)$

MAP / 1st order optim.

$$\arg\min_{x \in \mathbb{R}^n} f(x) + g(x)$$

✓ Converges.

✗ Unknown regularization g.

$$\begin{array}{c} D_\sigma \approx \mathrm{Id} - \nabla g \\ \hline D_\sigma \approx \mathrm{Prox}_{\tau g} \end{array}$$

Plug-and-Play Algorithms

✓ Implicit regularization.

✓ State-of-the-art performance.

✗ No convergence guarantees.

✗ No minimized functional.

✗ For a generic denoiser (e.g.: a deep neural net):

$$D_\sigma \neq \mathrm{Id} - \nabla g \text{ and } D_\sigma \neq \mathrm{Prox}_{\tau g}.$$

Objectif: Find sufficient conditions on the denoiser to ensure convergence of the PnP algorithms.

Existing works: fixed-point convergence

Show that PnP converges towards $x^* \in \mathrm{Fix}(T_{PnP})$.

$$x_{k+1} = \mathrm{Prox}_{\tau f} \circ (\tau D_\sigma + (1-\tau)\,\mathrm{Id})(x_k)$$
$$x_{k+1} = D_\sigma \circ (\mathrm{Id} - \tau \nabla f)(x_k)$$

Show that T_{PnP} is non-expansive:

- D_σ non-expansive (or averaged) [Sun et al. '19, Terris et al. '20, Hertrich et al. '21, Liu et al. '21, Bohra et al. '21]
$$||D_\sigma(x) - D_\sigma(y)|| \leq ||x - y||$$
✗ A performant generic deep denoiser is **not non-expansive**.

- data-fidelity term f strictly convex [Ryu et al., '19]
✗ Excludes numerous problems

PnP algorithms

Estimate x from observation $y \sim p(y|x)$

MAP / 1st order optim.

$$\arg\min_{x \in \mathbb{R}^n} f(x) + g(x)$$

✓ Converges.

✗ Unknown regularization g.

$$D_\sigma \approx \mathrm{Id} - \nabla g$$
$$D_\sigma \approx \mathrm{Prox}_{\tau g}$$

Plug-and-Play algorithms

✓ Implicit regularization.

✓ State-of-the-art performance.

✗ No convergence guarantees.

✗ No minimized functional.

D_σ non-expansif

Fixed-point convergence

$$x^* \in \mathrm{Fix}(T_{PnP})$$

✓ Convergence.

✗ Sub-optimal performance .

✗ No minimized functional.

Convergence via minimization

Prove that PnP algorithms converge towards $x^* \in \arg\min_x f(x) + g(x)$.

✗ For a generic denoiser (e.g.: a deep neural network):

$$D_\sigma \neq \mathrm{Id} - \nabla g \text{ et } D_\sigma \neq \mathrm{Prox}_g.$$

Solution: Make PnP real optimization algorithms by guaranteeing exactly

GradPnP algorithms	**ProxPnP algorithms**
$D_\sigma = \mathrm{Id} - \nabla g$	$D_\sigma = \mathrm{Prox}_g$

Convergence via minimization

Prove that PnP algorithms converge towards $x^* \in \arg\min_x f(x) + g(x)$.

✗ For a generic denoiser (e.g.: a deep neural network):

$$D_\sigma \neq \mathrm{Id} - \nabla g \text{ et } D_\sigma \neq \mathrm{Prox}_g.$$

Solution: Make PnP real optimization algorithms by guaranteeing exactly

GradPnP algorithms

$$D_\sigma = \mathrm{Id} - \nabla g$$

ProxPnP algorithms

$$D_\sigma = \mathrm{Prox}_g$$

Gradient-Step Denoiser [H., Leclaire, Papadakis, '21]

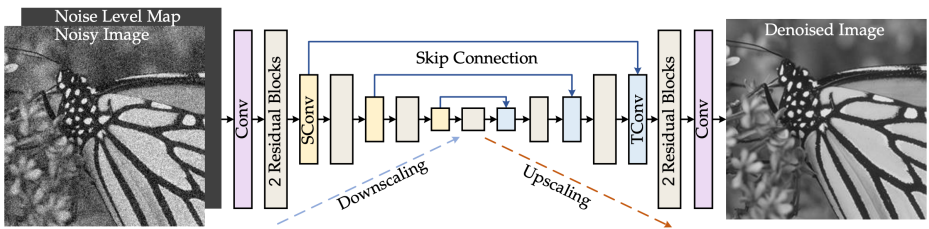
Gradient-Step Denoiser (GS) $D_\sigma = \mathrm{Id} - \nabla g_\sigma$

Gradient-Step Denoiser [H., Leclaire, Papadakis, '21]

Gradient-Step Denoiser (GS) $D_\sigma = \mathrm{Id} - \nabla g_\sigma$

- With the **non-convexe** potential: $g_\sigma(x) = \frac{1}{2}||x - N_\sigma(x)||^2$

$N_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ UNet with differentiable activations



[Zhang et al. '21]

$$D_\sigma(x) = x - \nabla\left(\frac{1}{2}||x - N_\sigma(x)||^2\right) = N_\sigma(x) + J_{N_\sigma}(x)^T(x - N_\sigma(x))$$

Gradient-Step Denoiser [H., Leclaire, Papadakis, '21]

Gradient-Step Denoiser (GS) $D_\sigma = \mathrm{Id} - \nabla g_\sigma$

- With the **non-convexe** potential: $g_\sigma(x) = \frac{1}{2}||x - N_\sigma(x)||^2$
- Trained by minimizing the L^2 cost

  ✓ **No degradation** of denoising performance

$\sigma(./255)$	15	25	50
UNet	33.90	31.25	28.00
GS-UNet	33.90	31.26	28.01

PSNR on the CBSD68 dataset

Gradient-Step Denoiser [H., Leclaire, Papadakis, '21]

Gradient-Step Denoiser (GS) $D_\sigma = \mathrm{Id} - \nabla g_\sigma$

- With the **non-convexe** potential: $g_\sigma(x) = \frac{1}{2}||x - N_\sigma(x)||^2$
- Trained by minimizing the L^2 cost

  ✓ **No degradation** of denoising performance

$\sigma(./255)$	15	25	50
UNet	33.90	31.25	28.00
GS-UNet	33.90	31.26	28.01

PSNR on the CBSD68 dataset

- The MMSE denoiser is optimal for the L^2 cost:

$$D_\sigma = \mathrm{Id} - \nabla g_\sigma \approx D_\sigma^{MMSE} = \mathrm{Id} - \nabla(-\sigma^2 \log p_\sigma)$$

✓ $g_\sigma \approx -\sigma^2 \log p_\sigma + \text{const}$

Nonconvex convergence of GradPnP algorithms [H., Leclaire, Papadakis, '21]

With GS denoiser, GradPnP algorithms converge towards a critical point of $f + \lambda g_\sigma$.

ex: GradPnP-PGD

$$\begin{aligned} x_{k+1} &= \mathrm{Prox}_{\tau f} \circ (\tau\lambda D_\sigma + (1 - \tau\lambda)\,\mathrm{Id})(x_k) \\ &= \mathrm{Prox}_{\tau f} \circ (\mathrm{Id} - \tau\lambda \nabla g_\sigma)(x_k) \end{aligned}$$

✗ g_σ is **non-convex**.

✓ g_σ has Lipschitz gradient

Hypotheses

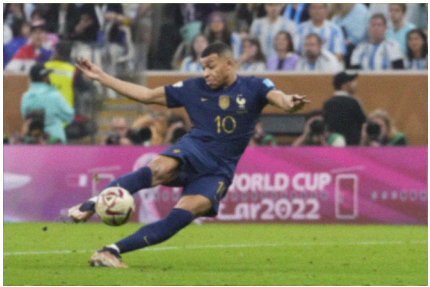
(i) f convex, proper, l.s.c

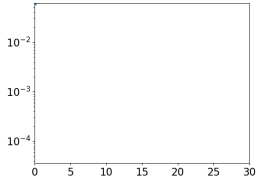
(ii) ∇g_σ L-Lipschitz ✓

(iii) $f + \lambda g_\sigma$ coercive and verifies the *Kurdyka-Lojasiewicz (KL)* property ✓

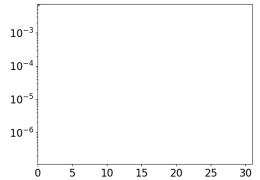
Then, for $\tau < \frac{2\lambda}{L}$, (x_k) converges towards a critical point of $f + \lambda g_\sigma$. [Attouch et al., '13]

Deblurring

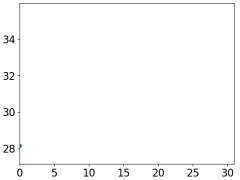




(a) $\frac{||x_{k+1}-x_k||}{||x_k||}$



(b) $F(x_k) - F^*$



(c) $\text{PSNR}(x_k)$

Experimental results

	Deblurring			Super-resolution		
Input noise level	0.01	0.03	0.05	0.01	0.03	0.05
Bicubic	-	-	-	24.85	23.96	22.79
EPLL [Zoran & Weiss, '11]	27.34	25.16	24.04	-	-	-
IRCNN [Zhang et al., '17]	31.42	28.01	26.40	26.97	25.86	25.45
DPIR [Zhang et al., '21]	**31.93**	**28.30**	26.82	27.79	26.58	25.83
GradPnP-PGD	31.70	28.28	**26.86**	27.88	**26.81**	**26.01**

PSNR (dB) pour deblurring and super-resolution tasks, averaged over the CBSD68 adtaset and over a variaty of blur kernels.

✓ Convergence + State-of-the-art performance among Plug-and-Play methods.

Convergence of PnP algorithms via minimization

Prouve that PnP algorithms converge towards $x^* \in \arg\min_x f(x) + g(x)$.

✗ For a generic denoiser (e.g.: a deep neural network):

$$D_\sigma \neq \mathrm{Prox}_{\tau g}.$$

Solution: Make PnP algorithms real optimization algorithms by guaranteeing:

GradPnP algorithms

$D_\sigma = \mathrm{Id} - \nabla g$

ProxPnP algorithms

$D_\sigma = \mathrm{Prox}_g$

Prox characterization

For an operator $T : \mathbb{R}^n \to \mathbb{R}^n$,

[Moreau, '65]

$T = \nabla h$ with h convex and T 1-Lipschitz
$\Leftrightarrow \exists \phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ **convex** s.t.

$$T = \mathrm{Prox}_{\phi}$$

✗ D_σ 1-Lipschitz non-realist.

Prox characterization

For an operator $T : \mathbb{R}^n \to \mathbb{R}^n$,

[Moreau, '65]

$T = \nabla h$ with h convex and T 1-Lipschitz
$\Leftrightarrow \exists \phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ **convex** s.t.

$$T = \mathrm{Prox}_{\phi}$$

✗ D_σ 1-Lipschitz non-realist.

[Gribonval et Nikolova, '20]

$T = \nabla h$ with h convex and T Lipschitz
$\Leftrightarrow \exists \phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$
$\left(1 - \frac{1}{\mathrm{Lip}(T)}\right)$ **weakly convex** s.t.

$$T = \mathrm{Prox}_{\phi}$$

✓ Verified by the MMSE denoiser

Prox characterization

For an operator $T : \mathbb{R}^n \to \mathbb{R}^n$,

[Moreau, '65]

$T = \nabla h$ with h convex and T 1-Lipschitz
$\Leftrightarrow \exists \phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ **convex** s.t.

$$T = \text{Prox}_{\phi}$$

✗ D_σ 1-Lipschitz non-realist.

[Gribonval et Nikolova, '20]

$T = \nabla h$ with h convex and T Lipschitz
$\Leftrightarrow \exists \phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$
$\left(1 - \frac{1}{\text{Lip}(T)}\right)$ **weakly convex** s.t.

$$T = \text{Prox}_{\phi}$$

✓ Verified by the MMSE denoiser

Proximal Denoiser

[H., Leclaire, Papadakis, '22]

Gradient-Step Denoiser:

$$D_\sigma = \text{Id} - \nabla g_\sigma = \nabla h_\sigma$$

with $h_\sigma = \frac{1}{2}||x||^2 - g_\sigma$

∇g_σ 1-Lipschitz $\Rightarrow$ h_σ convex

Corollary

If $\nabla g_\sigma = \text{Id} - D_\sigma$ is $(L \leq 1)$-Lipschitz
$\Rightarrow \exists \phi_\sigma : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$
$\frac{L}{L+1}$ weakly convex s.t.

$$D_\sigma = \text{Prox}_{\phi_\sigma}$$

$$\forall x \in \text{Im}(D_\sigma),\ \phi_\sigma = h_\sigma^*(x) - \frac{1}{2}||x||^2$$

How to ensure $\mathrm{Lip}(\nabla g_\sigma) \leq 1$? [H., Leclaire, Papadakis, '22]

Proximal Denoiser $D_\sigma = \mathrm{Id} - \nabla g_\sigma = \nabla h_\sigma = \mathrm{Prox}_{\phi_\sigma}$

- Convex parametrization of h_σ. [Amos et al., '17] [Goujon et al., '23]
  ✗ Suboptimal performance.
- Training cost penalization [Terris et. al, '21]:

$$\mathbb{E}_{x\sim p, y\sim\mathcal{N}(x,\sigma^2\,\mathrm{Id})}\left[||D_\sigma(y)-x||^2 + \mu\,\max(||\nabla^2 g_\sigma(y)||_S, 1-\epsilon)\right]$$

$\sigma(./255)$	5	15	25
GS-UNet	1.26	1.96	3.27
Prox-UNet	**0.92**	**0.99**	**0.96**

$\max_x ||\nabla^2 g_\sigma(x+\xi_\sigma)||_S$ on the CBSD68 dataset

$\sigma(./255)$	5	15	25
GS-UNet	**40.26**	**33.90**	**31.26**
Prox-UNet	40.12	33.60	30.82

PSNR on the CBSD68 dataset

Convergence of ProxPnP algorithms [H., Leclaire, Papadakis, '22]

With Prox denoiser, ProxPnP algorithms converge towards a critical point of $f + \lambda\phi_\sigma$.

ex: ProxPnP-PGD

$$x_{k+1} = D_\sigma \circ (\mathrm{Id} - \frac{1}{\lambda}\nabla f)(x_k)$$
$$= \mathrm{Prox}_{\phi_\sigma} \circ (\mathrm{Id} - \frac{1}{\lambda}\nabla f)(x_k)$$

⚠ $D_\sigma = \mathrm{Prox}_{\phi_\sigma} \neq \mathrm{Prox}_{\tau\phi_\sigma} \Rightarrow$ Requires to keep a stepsize $\tau = 1$.

Hypothesis

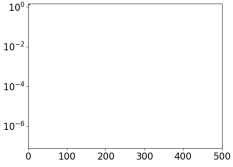
(i) f convex, differentiable, with L_f-Lipschitz gradient.

(ii) ϕ_σ M-weakly convex. ✓

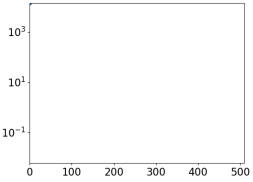
(iii) $f + \lambda\phi_\sigma$ coercive and verifies the *Kurdyka-Lojasiewicz (KL)* condition. ✓

Then, for $\frac{1}{\lambda}L_f + M < 2$, (x_k) converges towards a critical point of $f + \lambda\phi_\sigma$.

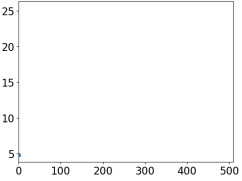
Super-resolution



(a) $\frac{||x_{k+1}-x_k||}{||x_k||}$



(b) $F(x_k) - F^*$



(c) $\text{PSNR}(x_k)$

Plug-and-Play algorithms

Estimate x from observation $y \sim p(y|x)$

MAP / 1st order optim.

$$\arg\min_{x \in \mathbb{R}^n} f(x) + g(x)$$

- ✓ Converging algorithms.
- ✗ Uknown regularization g.

$D_\sigma \approx \mathrm{Id} - \nabla g$

$D_\sigma \approx \mathrm{Prox}_{\tau g}$

Plug-and-Play algorithms

- ✓ Implicit regularization.
- ✓ State-of-the-art performance.
- ✗ No convergence guarantees.
- ✗ No minimized functional.

$D_\sigma = \mathrm{Id} - \nabla g_\sigma$

$D_\sigma = \mathrm{Prox}_{\phi_\sigma}$

D_σ non-expansif

GradPnP convergence

$$x^* \in \arg\min_{x \in \mathbb{R}^n} f(x) + g_\sigma(x)$$

ProxPnP convergence

$$x^* \in \arg\min_{x \in \mathbb{R}^n} f(x) + \phi_\sigma(x)$$

Fixed-point convergence

$$x^* \in \mathrm{Fix}(T_{PnP})$$

- ✓ Convergence.
- ✓ Optimize an explicit functional.
- ✓ State-of-the-art performance.

- ✓ Convergence.
- ✗ No minimized functional.
- ✗ Sub-optimal performance.

Bregman generalization and inverse problems with Poisson noise.

[H., Kamilov, Leclaire, Papadakis, '23]

Poisson inverse problems

If the observation noise is Poisson, $y \sim \mathcal{P}(Ax)$, the data-fidelity term is

$$f(x) = -\log p(y|x) = \mathrm{KL}(y, Ax)$$

✗ ∇f non-Lipschitz.

[Bauschke et. al, '17] For Burgs-entropy $h(x) = -\sum_i \log(x_i)$ and $L \geq ||y||_1$:

✓ **(L-smad)** $Lh - f$ convex on $int\,\mathrm{dom}(h)$.

- Generalizes the notion of **smoothness**:

$$\text{for } h(x) = \tfrac{1}{2}||x||^2, \text{ implies } \nabla f \ L\text{-Lipschitz.}$$

- h defines a **new geometry (Hessian manifold) adapted to the data-fidelity term** f:

Euclidian distance $\frac{1}{2}||x-y||^2$ replaced by the Bregman divergence
$$D_h(x, y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$$

[1] $D_h(x, y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$

Bregman potential	Euclidian $\frac{1}{2}\|\|x\|\|^2$	$h : \mathbb{R}^n \to \mathbb{R}$ strictly convex, $\mathcal{C}^2$
Distance	$\frac{1}{2}\|\|x-y\|\|^2$	Bregman Divergence $D_h(x,y)$ [1]
Smooth	∇f L-Lipschitz	$Lh - f$ convex

[1] $D_h(x,y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$

Bregman potential	Euclidian $\frac{1}{2}\|\|x\|\|^2$	$h : \mathbb{R}^n \to \mathbb{R}$ strictly convex, $\mathcal{C}^2$
Distance	$\frac{1}{2}\|\|x-y\|\|^2$	Bregman Divergence $D_h(x,y)$ [1]
Smooth	∇f L-Lipschitz	$Lh - f$ convex
Proximal operator	$\text{Prox}_f = \arg\min_x f(x) + \frac{1}{2}\|\|x-y\|\|^2$	$\text{Prox}_f^h = \arg\min_x f(x) + D_h(x,y)$

[1] $D_h(x,y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$

Bregman potential	Euclidian $\frac{1}{2}\|\|x\|\|^2$	$h : \mathbb{R}^n \to \mathbb{R}$ strictly convex, $\mathcal{C}^2$
Distance	$\frac{1}{2}\|\|x-y\|\|^2$	Bregman Divergence $D_h(x,y)$ [1]
Smooth	∇f L-Lipschitz	$Lh - f$ convex
Proximal operator	$\text{Prox}_f = \arg\min_x f(x) + \frac{1}{2}\|\|x-y\|\|^2$	$\text{Prox}_f^h = \arg\min_x f(x) + D_h(x,y)$
Gradient Desc. (GD)	$\text{Id} - \tau(\nabla f + \nabla g)$	$\nabla h^*(\nabla h - \tau(\nabla f + \nabla g))$
Prox. Grad. Desc. (PGD)	$\text{Prox}_{\tau g} \circ (\text{Id} - \tau \nabla f)$	$\text{Prox}_{\tau g}^h \circ \nabla h^*(\nabla h - \tau \nabla f)$

[H., Kamilov, Leclaire, Papadakis, '23]

[1] $D_h(x,y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$

Bregman potential	Euclidian $\frac{1}{2}\|\|x\|\|^2$	$h : \mathbb{R}^n \to \mathbb{R}$ strictly convex, $\mathcal{C}^2$
Distance	$\frac{1}{2}\|\|x-y\|\|^2$	Bregman Divergence $D_h(x,y)$ [1]
Smooth	∇f L-Lipschitz	$Lh - f$ convex
Proximal operator	$\text{Prox}_f = \arg\min_x f(x) + \frac{1}{2}\|\|x-y\|\|^2$	$\text{Prox}_f^h = \arg\min_x f(x) + D_h(x,y)$
Gradient Desc. (GD)	$\text{Id} - \tau(\nabla f + \nabla g)$	$\nabla h^*(\nabla h - \tau(\nabla f + \nabla g))$
Prox. Grad. Desc. (PGD)	$\text{Prox}_{\tau g} \circ (\text{Id} - \tau \nabla f)$	$\text{Prox}_{\tau g}^h \circ \nabla h^*(\nabla h - \tau \nabla f)$

[H., Kamilov, Leclaire, Papadakis, '23]

Noise model $p(y\|x)$	Gaussian	$\propto \exp\left(-\gamma D_h(x,y)\right)$

[1] $D_h(x,y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$

Bregman potential	Euclidian $\frac{1}{2}\|\|x\|\|^2$	$h : \mathbb{R}^n \to \mathbb{R}$ strictly convex, $\mathcal{C}^2$
Distance	$\frac{1}{2}\|\|x-y\|\|^2$	Bregman Divergence $D_h(x,y)$ [1]
Smooth	∇f L-Lipschitz	$Lh - f$ convex
Proximal operator	$\text{Prox}_f = \arg\min_x f(x) + \frac{1}{2}\|\|x-y\|\|^2$	$\text{Prox}_f^h = \arg\min_x f(x) + D_h(x,y)$
Gradient Desc. (GD)	$\text{Id} - \tau(\nabla f + \nabla g)$	$\nabla h^*(\nabla h - \tau(\nabla f + \nabla g))$
Prox. Grad. Desc. (PGD)	$\text{Prox}_{\tau g} \circ (\text{Id} - \tau \nabla f)$	$\text{Prox}_{\tau g}^h \circ \nabla h^*(\nabla h - \tau \nabla f)$

[H., Kamilov, Leclaire, Papadakis, '23]

Noise model $p(y\|x)$	Gaussian	$\propto \exp\left(-\gamma D_h(x,y)\right)$
MAP denoiser	$\text{Prox}_{-\sigma^2 \log p}(y)$	$\text{Prox}^h_{-\frac{1}{\gamma}\log p}(y)$
MMSE denoiser	$y + \sigma^2 \nabla \log p_\sigma(y)$	$y + \frac{1}{\gamma}\nabla^2 h(y)^{-1} \cdot \nabla \log p_\gamma(y)$

[1] $D_h(x,y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$

Bregman potential	Euclidian $\frac{1}{2}\|\|x\|\|^2$	$h : \mathbb{R}^n \to \mathbb{R}$ strictly convex, $\mathcal{C}^2$
Distance	$\frac{1}{2}\|\|x-y\|\|^2$	Bregman Divergence $D_h(x,y)$ [1]
Smooth	∇f L-Lipschitz	$Lh - f$ convex
Proximal operator	$\text{Prox}_f = \arg\min_x f(x) + \frac{1}{2}\|\|x-y\|\|^2$	$\text{Prox}_f^h = \arg\min_x f(x) + D_h(x,y)$
Gradient Desc. (GD)	$\text{Id} - \tau(\nabla f + \nabla g)$	$\nabla h^*(\nabla h - \tau(\nabla f + \nabla g))$
Prox. Grad. Desc. (PGD)	$\text{Prox}_{\tau g} \circ (\text{Id} - \tau \nabla f)$	$\text{Prox}_{\tau g}^h \circ \nabla h^*(\nabla h - \tau \nabla f)$

[H., Kamilov, Leclaire, Papadakis, '23]

Noise model $p(y\|x)$	Gaussian	$\propto \exp\left(-\gamma D_h(x,y)\right)$
MAP denoiser	$\text{Prox}_{-\sigma^2 \log p}(y)$	$\text{Prox}^h_{-\frac{1}{\gamma}\log p}(y)$
MMSE denoiser	$y + \sigma^2 \nabla \log p_\sigma(y)$	$y + \frac{1}{\gamma}\nabla^2 h(y)^{-1} \cdot \nabla \log p_\gamma(y)$
GradPnP algorithms: GD with ...	$\nabla g \leftarrow \text{Id} - D_\sigma$	$\nabla g(y) \leftarrow \nabla^2 h(y) \cdot (\text{Id} - D_\gamma)(y)$
ProxPnP algorithms: PGD with ...	$\text{Prox}_{\tau g} \leftarrow D_\sigma$	$\text{Prox}^h_{\tau g} \leftarrow D_\gamma$

[1] $D_h(x,y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$

Bregman potential	Euclidian $\frac{1}{2}\|\|x\|\|^2$	$h : \mathbb{R}^n \to \mathbb{R}$ strictly convex, $\mathcal{C}^2$
Distance	$\frac{1}{2}\|\|x-y\|\|^2$	Bregman Divergence $D_h(x,y)$ [1]
Smooth	∇f L-Lipschitz	$Lh - f$ convex
Proximal operator	$\mathrm{Prox}_f = \arg\min_x f(x) + \frac{1}{2}\|\|x-y\|\|^2$	$\mathrm{Prox}_f^h = \arg\min_x f(x) + D_h(x,y)$
Gradient Desc. (GD)	$\mathrm{Id} - \tau(\nabla f + \nabla g)$	$\nabla h^*(\nabla h - \tau(\nabla f + \nabla g))$
Prox. Grad. Desc. (PGD)	$\mathrm{Prox}_{\tau g} \circ (\mathrm{Id} - \tau \nabla f)$	$\mathrm{Prox}_{\tau g}^h \circ \nabla h^*(\nabla h - \tau \nabla f)$

[H., Kamilov, Leclaire, Papadakis, '23]

Noise model $p(y\|x)$	Gaussian	$\propto \exp\left(-\gamma D_h(x,y)\right)$
MAP denoiser	$\mathrm{Prox}_{-\sigma^2 \log p}(y)$	$\mathrm{Prox}^h_{-\frac{1}{\gamma}\log p}(y)$
MMSE denoiser	$y + \sigma^2 \nabla \log p_\sigma(y)$	$y + \frac{1}{\gamma}\nabla^2 h(y)^{-1} \cdot \nabla \log p_\gamma(y)$
GradPnP algorithms: GD with ...	$\nabla g \leftarrow \mathrm{Id} - D_\sigma$	$\nabla g(y) \leftarrow \nabla^2 h(y) \cdot (\mathrm{Id} - D_\gamma)(y)$
ProxPnP algorithms: PGD with ...	$\mathrm{Prox}_{\tau g} \leftarrow D_\sigma$	$\mathrm{Prox}^h_{\tau g} \leftarrow D_\gamma$
Gradient-Step Denoiser	$D_\sigma(y) = y - \nabla g_\sigma(y)$	$D_\gamma(y) = y - \nabla^2 h(y)^{-1} \cdot \nabla g_\gamma(y)$
Writes like a Prox *if ...*	h_σ convex	J_{D_γ} positive semi-definite

[1] $D_h(x,y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$

Application to Poisson inverse problems

Data-fidelity term $f(x) = KL(y, Ax)$

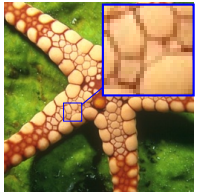
- **Bregman potential**: Burg's entropy $h(x) = -\sum_{i=1}^{n} \log(x_i)$

  $Lh - f$ convex for $L \geq ||y||_1$ [Bauschke, '17]

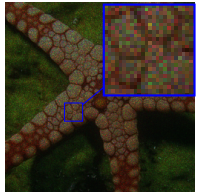
- **Bregman noise model**: Gamma Inverse ($\mathcal{IG}$) distribution

$$p(y|x) \propto \prod_{i=1}^{n} \left(\frac{x_i}{y_i}\right)^{\gamma} \exp\left(-\gamma \frac{x_i}{y_i}\right) = \prod_{i=1}^{n} \mathcal{IG}(\gamma - 1, \gamma x_i)$$

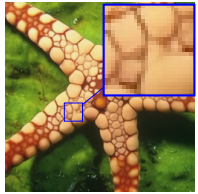
- **Gradient-Step Denoiser**: $D_\gamma(y) = y - y^2 \nabla g_\gamma(y)$



(a) Propre

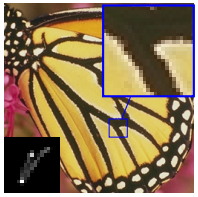


(b) noisy ($\gamma = 25$)

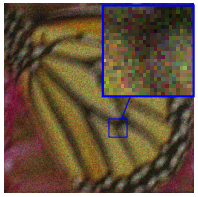


(c) Denoiser GS

Deblurring with Poisson noise

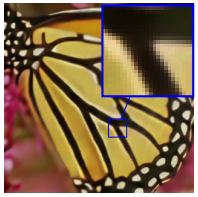


(a) Propre

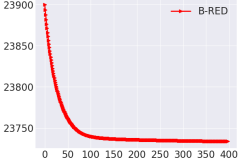


(b) Observation (16.21dB)

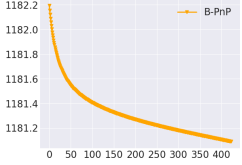
(c) GradPnP (23.01dB)



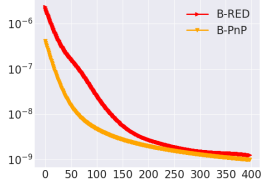
(d) ProxPnP (22.96dB)



(e) $(\lambda f + g_\gamma)(x_k)$ GradPnP



(f) $(\lambda f + \phi_\gamma)(x_k)$ ProxPnP



(g) $||x_{i+1} - x_i||^2$

Conclusion

Summary of the contributions

- Introduction of denoisers allowing to rewrite PnP algorithms like real optimization algorithms.
- Learning of an explicit approximation of the clean image prior.
- Convergence proofs towards critical points of an explicit and non-convex objective.
- State-of-the-art restoration performance.
- Bregman extension for non-Gaussian noises.

Open research questions

- **Convergence proofs with D_σ monotone but $D_\sigma \neq$ gradient**

Convergence of the PnP algorithm $D_\sigma \circ (\mathrm{Id} - \nabla f)$ towards the solution of a non-monotone inclusion problem.

- **How to relate the error when training a denoiser to the performance in Plug-and-Play / Diffusion models ?**

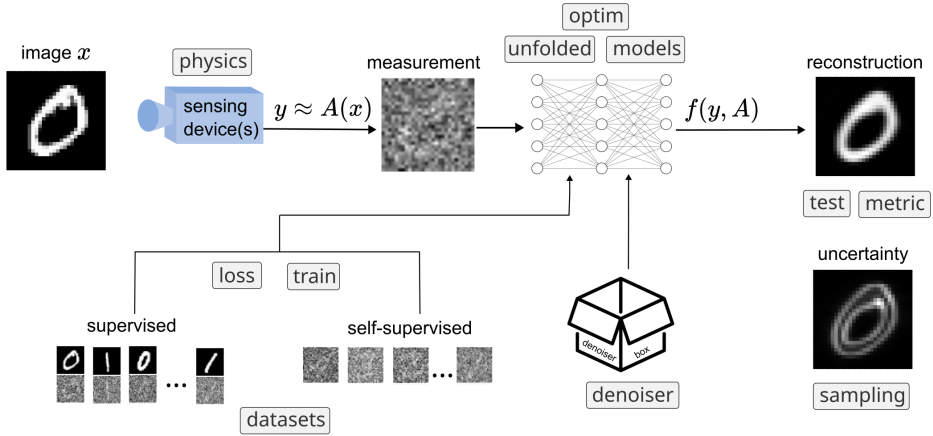
If the trained denoiser is ϵ-close to the optimal denoiser, do we still converge ? And if yes, to which point / distribution ?

- **Denoising and Optimal Transport**

The MMSE denoiser (optimal denoiser for the L_2 cost) writes as $D_\sigma = \nabla m_\sigma$ with m_σ convex. *Brenier theorem:* It is an optimal transport map from $p_\sigma = p * \mathcal{N}(0, \sigma^2 \,\mathrm{Id})$ to $D_\sigma \# p_\sigma$. How to relate $D_\sigma \# p_\sigma$ and p ?

Deep Inverse

```
pip install deepinv
```



Convergence proof with monotone denoiser

Objective: Generalize the nonconvex proof without requiring D_σ to write as a gradient.

Optimization	**Monotone inclusion**
Find $\arg\min_x f(x) + g(x)$	Find x, $0 \in A(x) + B(x)$
PGD : $x_{k+1} \in \text{Prox}_{\tau f} \circ (\text{Id} - \tau \nabla g)(x_k)$	$x_{k+1} \in J_{\tau A} \circ (\text{Id} - \tau B)(x_k)$
$T = \partial f$ and f convex $T = \nabla f$ and f convex and T L-Lipschitz $T = \text{Prox}_f$ f convex	T (maximally) monotone T $\left(\beta = \frac{1}{L}\right)$-cocoercive $T = J_A = (\text{Id} + A)^{-1}$, A monotone
$T = \partial f$ and f weakly convex $T = \text{Prox}_f$ and f weakly convex	T negative-monotone $T = J_A$, A negative-monotone

T monotone : $\langle x - y, T(x) - T(y) \rangle \geq 0$
T cocoercive : $\langle x - y, T(x) - T(y) \rangle \geq \rho ||T(x) - T(y)||^2$
T negative monotone : $\langle x - y, T(x) - T(y) \rangle \geq -\rho ||x - y||^2$ $(\rho \geq 0)$

[Gribonval et Nikolova, '20]

$T = \nabla h$ with h convex and T Lipschitz

$\Leftrightarrow \exists \phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\} \left(1 - \frac{1}{\mathrm{Lip}(T)}\right)$ **weakly convex** s.t.

$$T = \mathrm{Prox}_{\phi}$$

[Bauschke et. al , '21]

T β-cocoercive

$\Leftrightarrow \exists A : \mathbb{R}^n \to \mathbb{R}^n$ $(\rho = 1 - \beta)$ **negative-monotone** s.t.

$$T = J_A = (\mathrm{Id} + A)^{-1}$$

T β-cocoercive $\Leftrightarrow \frac{2}{\beta} T - \mathrm{Id}$ 1-Lipschitz

Objectives : Convergence of the PnP algorithm $D_\sigma \circ (\mathrm{Id} - \nabla f)$ towards the solution of a non-monotone inclusion problem.

- Study the cocoercive coefficient of state-of-the-art denoiser $\Rightarrow$ resolvents of negative monotone operators.
- Convergence proof of the forward-backward algorithm for non-monotone inclusion problems ?

Convergence of ProxPnP algorithms [H., Leclaire, Papadakis, '22]

With Prox denoiser, ProxPnP algorithms converge towards a critical point of $f + \lambda\phi_\sigma$.

ex: ProxPnP-PGD

$$x_{k+1} = D_\sigma \circ (\mathrm{Id} - \frac{1}{\lambda}\nabla f)(x_k)$$
$$= \mathrm{Prox}_{\phi_\sigma} \circ (\mathrm{Id} - \frac{1}{\lambda}\nabla f)(x_k)$$

⚠ $D_\sigma = \mathrm{Prox}_{\phi_\sigma} \neq \mathrm{Prox}_{\tau\phi_\sigma} \Rightarrow$ Requires to keep a stepsize $\tau = 1$.

Hypothesis

(i) f convex, differentiable, with L_f-Lipschitz gradient.

(ii) ϕ_σ M-weakly convex. ✓

(iii) $f + \lambda\phi_\sigma$ coercive and verifies the *Kurdyka-Lojasiewicz (KL)* condition. ✓

Then, for $\frac{1}{\lambda}L_f + M < 2$, (x_k) converges towards a critical point of $f + \lambda\phi_\sigma$.

Convergence of ProxPnP algorithms [H., Leclaire, Papadakis, '22]

With Prox denoiser, ProxPnP algorithms converge towards a critical point of $f + \lambda\phi_\sigma$.

ex: ProxPnP-PGD

$$x_{k+1} = D_\sigma \circ (\mathrm{Id} - \frac{1}{\lambda}\nabla f)(x_k)$$
$$= \mathrm{Prox}_{\phi_\sigma} \circ (\mathrm{Id} - \frac{1}{\lambda}\nabla f)(x_k)$$

⚠ $D_\sigma = \mathrm{Prox}_{\phi_\sigma} \neq \mathrm{Prox}_{\tau\phi_\sigma} \Rightarrow$ Requires to keep a stepsize $\tau = 1$.

Hypothesis

(i) f convex, differentiable, with L_f-Lipschitz gradient.

(ii) ϕ_σ M-weakly convex. ✓

(iii) $f + \lambda\phi_\sigma$ coercive and verifies the *Kurdyka-Lojasiewicz (KL)* condition. ✓

Then, for $\boxed{\frac{1}{\lambda}L_f + M < 2}$ (x_k) converges towards a critical point of $f + \lambda\phi_\sigma$.

✗ Regularisation parameter λ lower-bounded: suboptimal.

Relaxed PnP-αPGD algorithm [H., Chambolle, Leclaire, Papadakis, '23]

Solution: Inertial relaxation of the PGD algorithm (αPGD):

$$\text{Pour } 0 < \alpha < 1 \quad \begin{cases} q_{k+1} &= (1-\alpha)y_k + \alpha x_k \\ x_{k+1} &= D_\sigma^\gamma \circ (x_k - \frac{1}{\lambda}\nabla f(q_{k+1})) \\ y_{k+1} &= (1-\alpha)y_k + \alpha x_{k+1}. \end{cases}$$

with proximal denoiser γ**-relaxed** $D_\sigma^\gamma = \mathrm{Id} - \gamma \nabla g_\sigma = \mathrm{Prox}_{\phi_\sigma^\gamma}$

✓ ϕ_σ^γ is $\frac{\gamma L}{\gamma L + 1}$ weakly convex: **control** of the weak convexity constant γ.

Hypothesis

(i) f convex, differentiable, with gradient L_f-Lipschitz.

(ii) ϕ_σ^γ M_γ-weakly convex. ✓

Then, for $\frac{1}{\lambda} L_f M_\gamma < 1$ and $M_\gamma < \alpha < \frac{\lambda}{L_f}$, ✓ Allow λ small when $\gamma \to 0$

(x_k) converges towards a critical point of $f + \lambda \phi_\sigma^\gamma$.

Kurdyka-Lojasiewicz property

Definition (Kurdyka-Łojasiewicz (KŁ) property [Attouch et. al '10])

A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is said to have the Kurdyka-Łojasiewicz property at $x^* \in dom(f)$ if there exists $\eta \in (0, +\infty)$, a neighborhood U of x^* and a continuous concave function $\phi : [0, \eta) \to \mathbb{R}_+$ such that $\phi(0) = 0$, ϕ is $\mathcal{C}^1$ on $(0, \eta)$, $\phi' > 0$ on $(0, \eta)$ and $\forall x \in U \cap [f(x^*) < f < f(x^*) + \eta]$, the Kurdyka-Łojasiewicz inequality holds:

$$\phi'(f(x) - f(x^*))\text{dist}(0, \partial f(x)) \geq 1. \tag{1}$$

Proper lsc functions that satisfy the Kurdyka-Łojasiewicz inequality at each point of $dom(\partial f)$ are called KŁ functions.

For f smooth and $f(x^*) = 0$, the KŁ inequality equation 1 writes $||\nabla(\phi \circ f)(x)|| \geq 1$
Useful set of functions with stability properties which verify the KŁ property:

- Real analytic functions
- Semialgebraic functions
- Subanalytic functions [Bolte et. al, '07]

Id $-D_\sigma$ vs D_σ non-expansif

σ(./255)	5	10	15	20	25
GS-DRUNet	40.27	36.16	33.92	32.41	31.28
prox-DRUNet	40.04	35.86	33.51	31.88	30.64
nonexp-DRUNet	34.92	32.90	31.42	30.30	29.42

PSNR (dB) de Débruitage on the CBSD68 dataset, pour différents niveaux de noises σ.

Performance numérique des PnP algorithms avec Denoiser proximal

$\sigma(./255)$	2.55	7.65	12.75
IRCNN	31.42	28.01	26.40
DPIR	31.93	28.30	26.82
GSRED-PGD	31.70	28.28	26.86
ProxPnP-PGD	30.91	27.97	26.66
ProxPnP-αPGD	31.55	28.03	26.66
ProxPnP-DRSdiff	30.56	27.78	26.61
ProxPnP-DRS	31.51	28.01	26.63
nonexp-PnP-PGD	30.25	27.06	25.30

PSNR (dB) lors du defloutage on the CBSD68 dataset, moyennés pour différents noyaux de flous

Proprietés de g_σ

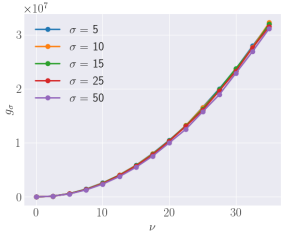
- g_σ **coercive:**

On choisir C compact convexe ($C = [-1, 2]^n$) et

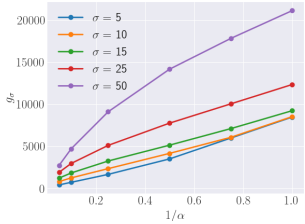
$$\hat{g}_\sigma(x) = g_\sigma(x) + \frac{1}{2}||x - \Pi_C(x)||^2 = \frac{1}{2}||x - N_\sigma(x)||^2 + \frac{1}{2}||x - \Pi_C(x)||^2$$

où Π_C la projection euclidienne sur C.

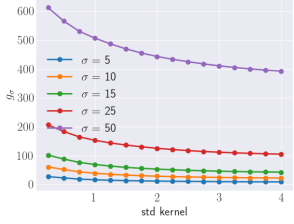
- g_σ **sous-analytique**



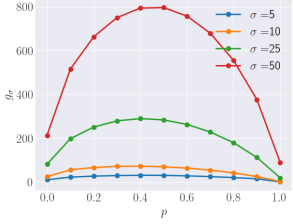
(a) Gaussian noise $y \sim \mathcal{N}(x, \nu^2\,\mathrm{Id})$



(b) Poisson noise $y \sim \frac{1}{\alpha}\mathcal{P}(\alpha x)$



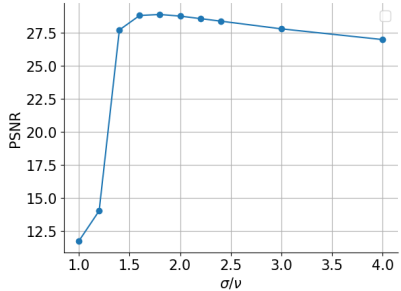
(c) Gaussian blur $y = k_\nu * x$

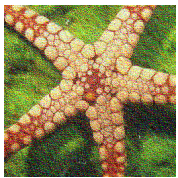


(d) Missing pixels $y = M_p x$

Evolution, for various σ, of $g_\sigma(y)$ where y has been obtained from x with different degradations.

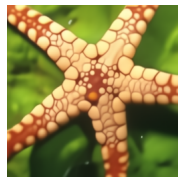
Influence of the parameter σ





$\sigma/\nu = 1$

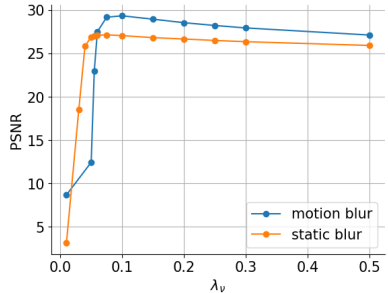
$\sigma/\nu = 1.8$

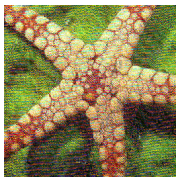


$\sigma/\nu = 4$

$\sigma/\nu = 10$

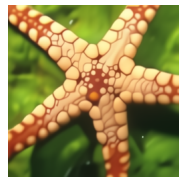
Influence of the parameter λ



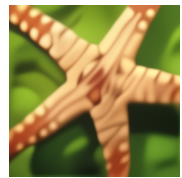


$\lambda\nu^2 = 1$

$\lambda\nu^2 = 1.8$



$\lambda\nu^2 = 4$



$\lambda\nu^2 = 10$

Fixed-points of the GS-Denoiser

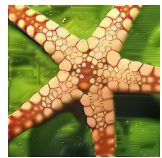
$$x_{k+1} = \mathcal{P}_{[0,1]^n}(x_k - \tau \nabla g_\sigma(x_k))$$

(a) init

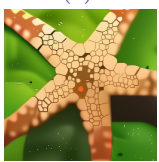
(b) $\sigma = 0$

(c) $\sigma = 0.1$

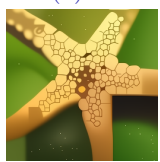
(d) $\sigma = 0.5$



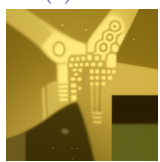
(e) $\sigma = 1$



(f) $\sigma = 5$

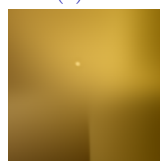


(g) $\sigma = 10$



(h) $\sigma = 20$

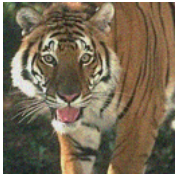
(i) $\sigma = 30$



(j) $\sigma = 40$

Débruitage avec N_σ

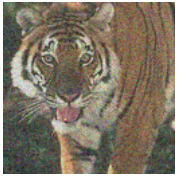
$$D_\sigma(x) = x - \nabla\left(\frac{1}{2}||x - N_\sigma(x)||^2\right) = N_\sigma(x) + J_{N_\sigma}(x)^T(x - N_\sigma(x))$$



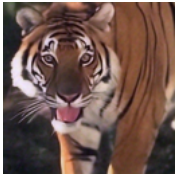
$\sigma = 10$

N_σ (29.16dB)

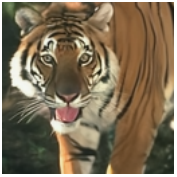
D_σ (35.55dB)



$\sigma = 25$



N_σ (23.94dB)

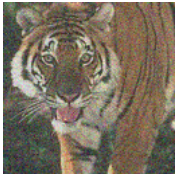


D_σ (30.49dB)

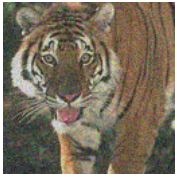
Denoiser relâché $D_\sigma^\gamma = \mathrm{Id} - \gamma \nabla g_\sigma$

Le Denoiser is entraîné pour que $\nabla g_\sigma(x + \sigma n) \approx \sigma n$ où $n \sim \mathcal{N}(0, \mathrm{Id})$.

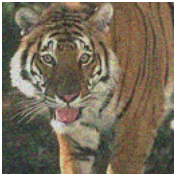
$$D_\sigma^\gamma(x + \sigma n) \approx x + (1 - \gamma)\sigma n$$



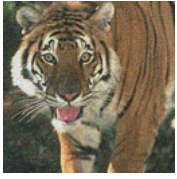
$\gamma = 0$



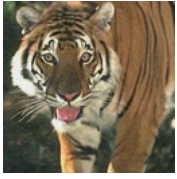
$\gamma = 0.1$



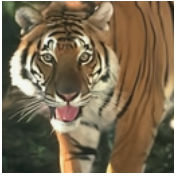
$\gamma = 0.25$



$\gamma = 0.5$



$\gamma = 0.75$



$\gamma = 1$

Proximal Denoiser [Hurault, Leclaire, Papadakis, '22]

Proximal Denoiser $D_\sigma = \mathrm{Id} - \nabla g_\sigma = \mathrm{Prox}_{\phi_\sigma}$

$$\nabla g_\sigma \ L < 1 \text{ Lipschitz} \Rightarrow D_\sigma \textbf{ invertible}$$

From $y = D_\sigma(x)$, one can recover x via fixed-point iterations

$$x_{k+1} = \nabla g_\sigma(x_k) + y$$

x

$D_\sigma(x)$

x_∞

PnP fixed-point convergence

$$x_{k+1} = T_{PnP}(x_k)$$

$$\text{with } T_{PnP} = \begin{cases} T_{HQS} & = D_\sigma \circ \operatorname{Prox}_{\tau f} \\ T_{PGD} & = D_\sigma \circ (\operatorname{Id} - \tau \nabla f) \\ T_{DRS} & = \frac{1}{2}\operatorname{Id} + \frac{1}{2}(2D_\sigma - \operatorname{Id}) \circ (2\operatorname{Prox}_{\tau f} - \operatorname{Id}) \end{cases}$$

Objective: Show that $x_k \to x^* \in \operatorname{Fix}(T_{PnP})$.

Solution: Averaged (nonexpansive) D_σ. [Sun et. al '18, '19, '21] [Hertrich et. al '20] [Terris et. al '21] [Bohra et. al '21]

- For $\theta \in (0,1)$, $T : \mathbb{R}^n \to \mathbb{R}^n$ θ-averaged if $T = \theta R + (1-\theta)\operatorname{Id}$ for some R nonexpansive (1-Lipschitz).
- If T is a θ-averaged operator that admits fixed points, then the sequence $x_{k+1} = T(x_k)$ converges to a fixed point of T
- If D_σ is averaged then T_{PnP} is averaged for all presented PnP operators.

Non-expansivity harms denoising performance.

Self-Consistent Velocity Matching of Probability Flows [Li, H., Solomon '23]

- We takle PDEs of the form

$$\frac{\partial \mu_t}{\partial t}(x) = -\operatorname{div}\left(\mu_t f_t(x, \mu_t)\right) \tag{2}$$

  where μ_t is a probability density function. The Wasserstein Gradient Flow corresponds to $f_t(x, \mu) = -\nabla_{W_2} D(\mu_t, \pi)(x)$.

- μ_t and its velocity field v_t are linked via the **continuity equation**

$$\frac{\partial \mu_t}{\partial t} = -\operatorname{div}\left(\mu_t v_t\right). \tag{3}$$

- If a probability flow μ_t with velocity field v_t satisfies the following fixed-point equation

$$v_t(x) = f_t(x, \mu_t) \tag{4}$$

  then the PDE is solved.

We parameterize v_t^θ with a time-dependent neural network and minimize

$$\min_\theta \int_0^T \mathbb{E}_{X \sim \mu_t^\theta}\left[||v_t^\theta(X) - f_t(X, \mu_t^\theta)||^2\right] dt. \tag{5}$$