PnP-Flow : Plug-and-Play Image Restoration with Flow Matching

Bissmella Bahaduri
MVA, ENS Paris-Saclay
bissmellabahaduri@gmail.com

Gaëtan Patinier
MVA, ENS Paris-Saclay
gaetan.patinier@eleves.enpc.fr

1 Introduction

Image restoration is an inverse problem that aims at discovering an image x from a noisy, degraded given image y. Given an image x sampled from a random variable X with distribution P_X, and an observation y from a random variable Y with distribution P_Y such that $y = \text{noisy}(Hx)$, where H is a degradation operator and noisy is a noise model, we aim to find an image x with highest a posteriori value:

$$\arg\max_x \{\log P_X(X = x|Y = y)\} = \arg\max_x \{\log P_Y(Y = y|X = x) + \log P_X(x)\} \tag{1}$$

This is reformulated as:

$$\arg\min_x \{F(x) + R(x)\} \tag{2}$$

where the first term in the formula above is the data-fidelity term $F(x) = -\log P_Y(Y = y|X = x)$ and the second term is the regularizer $R(x) = -\log P_X(x)$. While the first term is optimized using gradient descent, the second term is optimized using proximal splitting method. More specifically, current methods use neural network based denoisers as priors for the regularizer term.

Recently, generative models have seen many advancements and are being widely used for image generation.

Flow Matching models are generative models which learn a velocity field $v : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ that defines a continuous transformation from a latent distribution to the target data distribution. Once $v_t(x)$ is learned, sampling from the target distribution is achieved by solving the following ordinary differential equation (ODE):

$$\frac{\partial f(t, x)}{\partial t} = v_t(f(t, x)), \quad f(0, x) = x, \tag{3}$$

where x is sampled from the latent distribution.

The function $f(t, x)$ represents the trajectory of a data point evolving under the velocity field v_t. By integrating this ODE from $t = 0$ to $t = 1$, we obtain samples from the data distribution.

Flow Matching models differ from diffusion models in that they rely on deterministic ODEs rather than stochastic differential equations (SDEs), allowing for efficient sampling without requiring stochastic noise. This makes them computationally appealing for generative modeling tasks.

However, using flow matching methods for image restoration is not an easy task mainly due to the complications posed by the change of variables from the MAP (Maximum a Posteriori).

Instead, Martin et al. (2025) proposes to simply use a PnP (Plug-and-Play) method by plugging a FM (Flow Matching) model as a denoiser in a forward-backward algorithm.

In the following sections, we first provide a brief context on both flow matching and PnP methods, we then explain the proposed methodology in Martin et al. (2025).

2 Context

2.1 Background on PnP and Proximal Splitting

Plug-and-Play (PnP) algorithms extend classical proximal splitting methods, such as Forward-Backward Splitting (FBS) and the Alternating Direction Method of Multipliers (ADMM) by replacing explicit regularization terms with learned denoising priors. These methods solve optimization problems of the form 2.

The Proximal Operator A key component of proximal splitting methods is the proximal operator associated with the regularization function $R(x)$, defined as:

$$\text{prox}_{\gamma R}(y) := \arg\min_{x \in \mathbb{R}^d} \frac{1}{2}\|y - x\|^2 + \gamma R(x), \tag{4}$$

where $\gamma > 0$ is a step size parameter. This operator balances proximity to y with minimization of $R(x)$. In PnP algorithms, the proximal operator is often replaced by a learned denoiser, enabling the use of deep priors.

Forward-Backward Splitting (FBS) We focus on the Forward-Backward Splitting (FBS) algorithm, an iterative method that alternates between:

1. A gradient descent step on $F(x)$:
$$x^{k+1/2} = x^k - \gamma \nabla F(x^k). \tag{5}$$
2. A proximal step on $R(x)$:
$$x^{k+1} = \text{prox}_{\gamma R}(x^{k+1/2}). \tag{6}$$

We assume that $F(x)$ has a Lipschitz-continuous gradient with Lipschitz constant $\text{Lip}(\nabla F)$. The convergence of FBS to a minimizer of $F + R$ is guaranteed if the step size is chosen as:

$$\gamma \in \left(0, \frac{1}{\text{Lip}(\nabla F)}\right). \tag{7}$$

In PnP-FBS, the proximal step is replaced with a deep denoiser, implicitly imposing learned priors on the solution.

A main drawback of this method is that constraining the network's Lipschitz constant hurts the expressiveness of the network.

2.2 Flow Matching

Flow Matching is a method for learning a continuous transformation from a *latent distribution* (e.g., Gaussian noise) to a *target distribution* (e.g., real data). Instead of performing this transformation in a single step, Flow Matching constructs a smooth interpolation between the two distributions.

Let P_0 and P_1 denote the latent and target probability distributions respectively. A *coupling* π between P_0 and P_1 represents a joint distribution over (x_0, x_1), where $x_0 \sim P_0$ and $x_1 \sim P_1$.

Rather than transporting samples directly, we define a continuous probability path P_t interpolating between P_0 and P_1:

$$P_t := (1 - t)x_0 + tx_1, \quad t \in [0, 1], \tag{8}$$

where $x_0 \sim P_0$ and $x_1 \sim P_1$. The path P_t is absolutely continuous and satisfies:

$$\partial_t P_t + \nabla \cdot (P_t v_t) = 0. \tag{9}$$

Here, $v_t(x)$ is a *velocity field* that describes how points evolve over time.

Given a velocity field $v_t(x)$, the transformation follows the ordinary differential equation (ODE):

$$\partial_t f(t,x) = v_t(f(t,x)), \quad f(0,x) = x. \tag{10}$$

If f is known, we can sample from P_1 by first drawing $x_0 \sim P_0$ and computing $x_1 = f(1, x_0)$.

In practice, we approximate the velocity field $v_t(x)$ by training a neural network $v_\theta(t,x)$ with parameters θ. The objective is to minimize the *Flow Matching loss*:

$$L_{\text{FM}}(\theta) = \mathbb{E}_{t\sim U[0,1], x\sim P_t} \|v_\theta(t,x) - v_t(x)\|^2. \tag{11}$$

Since the true velocity field $v_t(x)$ is not available, we instead use the *Conditional Flow Matching (CFM) loss*:

$$L_{\text{CFM}}(\theta) = \mathbb{E}_{t\sim U[0,1], (x_0,x_1)\sim\pi} \|v_\theta(t, e_t(x_0,x_1)) - (x_1 - x_0)\|^2. \tag{12}$$

This loss function ensures that the instantaneous velocity remains proportional to the total displacement along the trajectory and only requires sampling from the coupling π, making it more practical.

Straight-Line Flows A common choice for the interpolation path is the straight-line flow:

$$X_t = (1-t)X_0 + tX_1. \tag{13}$$

This approach is computationally efficient because it represents the shortest path between two points and requires fewer steps in an ODE solver. Recent works Kornilov et al. (2024); Lee et al. (2023); Yang et al. (2024) have explored straight-line flows to improve sampling efficiency.

3 Flow matching as denoiser in PnP

Given $X_0 \sim P_0$ and $X_1 \sim P_1$ with joint probability $(X_0, X_1) \sim \pi$ and a pre-trained velocity field v_θ, the following time-dependent denoiser is defined for $t \in [0,1]$:

$$D_t := \text{Id} + (1-t)v_\theta(t,\cdot). \tag{14}$$

We propose to use this denoiser within the Plug-and-Play (PnP) framework.

For a fixed time $t \in [0,1]$, the minimizer v_t^* of the Conditional Flow Matching (CFM) loss over all possible vector fields is:

$$v_t^*(x) = \mathbb{E}[X_1 - X_0 | X_t = x],$$

where $X_t := e_t(X_0, X_1) = (1-t)X_0 + tX_1$. Assuming the ideal case where $v_\theta(t,x) = v_t^*(x)$, the denoiser becomes:

$$D_t(x) = x + (1-t)v_t^*(x) = \mathbb{E}[X_1 | X_t = x].$$

The operator D_t minimizes the L_2 distance between the real data and a projected data point on the trajectory at time t:

$$\min_g \mathbb{E}[(X_1 - g(X_t))^2],$$

projecting any noisy point taken along the path onto the target distribution. This shows that D_t acts as a denoiser, approximating the target distribution P_1 from noisy samples along the flow path.

The best denoising performance is achieved when using straight-line flows. In particular, for straight-line flows, the denoiser can efficiently project samples from the latent space to the data distribution. This is consistent with the recent work by **??**, where straight-line flows are preferred for computational efficiency and improved denoising.

3.1 Proposition 1: Perfect Denoising and Straight-Line Flow Matching

A main distinction between the flow matching and diffusion models is the trajectory taken from the latent to sampled data. Unlike flow matching, in diffusion models, the path is not straight.

3.2 PnP based on flow matching

In this section, we detail the algorithm proposed by Martin et al. (2025) that incorporates the denoiser D_t desecribed in 14 into a Forward-Backward Splitting (FBS) algorithm to solve inverse problems. The proposed algorithm has two main distinctions from the conventional PnP-BFS methods:

1. The algorithm's iterations depend on time due to the time-varying denoiser D_t.
2. An intermediate reprojection step is introduced between the gradient and denoising steps.

The updates at each time $t \in [0, 1]$ are as follows:

1. **Gradient step:** A gradient step on the data-fidelity term is applied, mapping x to $z = x - \gamma \nabla F(x)$, where $\gamma > 0$ is the learning rate.
2. **Interpolation step:** Since the operator D_t is designed to work best when the data point lies along the straight-line path $X_t = (1-t)X_0 + tX_1$, if the output z from the gradient step does not lie on this path, an interpolation is performed to ensure the denoiser works effectively. The interpolation is defined as: $\tilde{z} = (1-t)\epsilon + tz$ where ϵ is a noise sample drawn from P_0.
3. **PnP Denoising step:** The denoiser D_t is applied to the interpolated point $\tilde{z}$, regularizing the current image by pushing it toward the target distribution P_1.

Following is a detailed explanation of the algorithm:

PnP Flow Matching Algorithm Explanation:

1. **Input:**
   - Pre-trained velocity field v_θ from Flow Matching.
   - Time sequence $(t_n)_n$ with $t_n \in [0, 1]$.
   - Adaptive step sizes $(\gamma_n)_n$.
2. **Initialization:**
   - Start with an initial guess $x_0 \in \mathbb{R}^d$ (could be an image or data point).
3. **Main Iteration (Loop over each step):** For each $n = 0, 1, \ldots$, perform the following:
   (a) **Gradient Step:** Compute the gradient descent update based on the data fidelity term:
   $$z_n = x_n - \gamma_n \nabla F(x_n)$$
   (b) **Linear Interpolation:** Interpolate between the current gradient step z_n and a sample from the latent distribution $\epsilon \sim P_0$:
   $$\tilde{z}_n = (1 - t_n)\epsilon + t_n z_n$$
   (c) **Denoising Step:** Apply the denoising operator D_{t_n} to the interpolated value $\tilde{z}_n$:
   $$x_{n+1} = D_{t_n}(\tilde{z}_n)$$
4. **Output:** After iterating, the final estimate is x_{n+1}.

3.3 Techniques for denoising steps and learning rate

Averaging in the Denoising Step: Instead of using a single noise realization in the denoising step, we can average over multiple noise samples, $\epsilon \sim P_0$. This is done in the following way:

$$x_{n+1} := \mathbb{E}_{\epsilon \sim P_0} \left[D_{t_n} \left(\tilde{z}_n(\epsilon) \right) \right]$$

where $\tilde{z}_n(\epsilon) = (1 - t_n)\epsilon + t_n z_n$, and D_{t_n} is the denoising operator at time t_n. By averaging over multiple noise realizations, the output becomes more stable and less sensitive to randomness.

Time Dependent Learning Rate: A constant learning rate may lead to overly strong reliance on the data fit. For instance, with a constant learning rate $\gamma_t = 1$, the algorithm would return the noisy sample y as $D_1 = I_d$, which results in no denoising. To avoid this, the learning rate should decrease over time to balance the contributions of the data fit and denoiser.

The learning rate is set by a scheduler:

$$\gamma_t = (1 - t)^\alpha$$

where $\alpha \in (0, 1]$. This decay ensures that as time progresses, the denoising step becomes more important.

4 Performance on 2 dimensional data

In order to further test the proposed algorithm, we did an experiment with a small MLP model and 2D dataset.

First we trained the flow matching model following same method as Tong et al. (2023) using CFM loss on transforming 2D data points from normal distribution to moon distribution. An example output of the trained model can be seen in figure 1. It is obvious that the flow trajectory between the two distributions are straight lines. The result of the PnP Flow Matching on restoration from a disturbance of setting one or both of the dimensions to zeros for part of the data can be seen in figure 2.

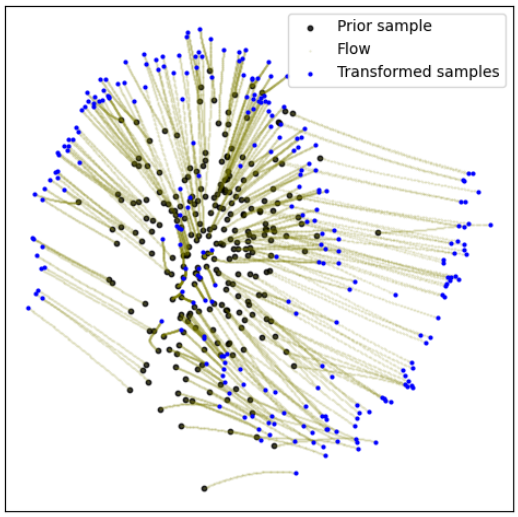

Figure 1: Transforming 2D data from normal distribution to moon distribution by FM model.

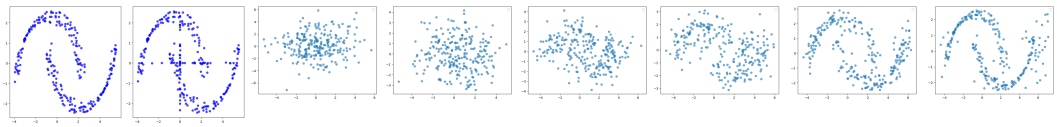

Figure 2: PnP-FM algorithm result on disturbed data. 1st left: true data; 2nd left: disturbed data.

5 Convergence of the Algorithm

In the context of the provided algorithm, we assume that the sequence of updates generated by the algorithm remains bounded over time. This means that the sequence of points does not grow indefinitely, and we expect it to converge to a limit as the algorithm proceeds.

Proposition 1. *Assume that $F : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable and that the learned vector field $v : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is continuous. Let the time sequence $(t_n)_{n\in\mathbb{N}}$ satisfy $\sum_{n=0}^{\infty}(1-t_n) < +\infty$ and let $\gamma_n := 1 - t_n$, for $n \in \mathbb{N}$. If the sequence $(x_n)_{n\in\mathbb{N}}$ obtained by the algorithm is bounded, then it converges.*

Proof: By the algorithm and the definition of D_{t_n}, we have:

$$x_{n+1} = D_{t_n}(u_n) = u_n + (1 - t_n)v_\theta(t_n)(u_n),$$

where, together with the definition of γ_n, we have:

$$u_n := (1 - t_n)\epsilon + t_n \left(x_n - \gamma_n \nabla F(x_n)\right) = t_n x_n + (1 - t_n)\left(\epsilon - t_n \nabla F(x_n)\right).$$

Thus, we obtain:

$$\|x_{n+1} - x_n\| = (1 - t_n)\|\epsilon - x_n - t_n \nabla F(x_n) + v_\theta(t_n)(u_n)\|.$$

By assuming that ∇F and v are continuous, and since (x_n) is bounded, the expression inside the norm is also bounded by some constant $M > 0$. Therefore, we conclude:

$$\sum_{n=0}^{\infty} \|x_{n+1} - x_n\| = M \sum_{n=0}^{\infty}(1 - t_n) < \infty,$$

which implies that the sequence (x_n) is a Cauchy sequence and hence converges.

6 Determinism and Convergence of the Algorithm

In the PnP Flow Matching algorithm, **determinism** is influenced by the noise ϵ drawn from the distribution P_0 (the latent distribution). The key observation is that ϵ is a random variable drawn at each iteration. It means that certain intermediate steps are inherently stochastic. Therefore, while the algorithm follows a deterministic update rule for x_n given $\tilde{z}_n$, the noise component introduces randomness into the algorithm's behavior.

Thus, the algorithm is **not fully deterministic** because of the noise sampling during the interpolation step. However, the overall convergence result still holds because the sequence of iterates remains bounded, and the randomness in the noise is averaged over multiple noise samples.

If one desires a **fully deterministic** version of the algorithm, it would be possible to replace the noise ϵ with a fixed value, such as the mean of the latent distribution P_0. However, this modification would likely alter the behavior and performance of the algorithm, especially in the context of denoising and image restoration tasks, where randomness often helps to avoid overfitting and provides generalization.

6.1 Reformulation of Proposition 4: Convergence of PnP Flow Matching

In light of the discussion on the stochastic nature of the algorithm, we propose the following reformulation of Proposition 4, considering the convergence of the PnP Flow Matching algorithm.

Proposition 2. *Let the operator D_t represent the denoising step defined in the PnP Flow Matching algorithm. Assume that the objective function $F : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable, and the learned vector field $v : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is continuous. Additionally, let the time sequence $(t_n)_{n\in\mathbb{N}}$ satisfy*

$$\sum_{n=0}^{\infty}(1 - t_n) < +\infty,$$

with the adaptive learning rate $\gamma_n = 1 - t_n$ for all $n \in \mathbb{N}$.

Then, if the sequence $(x_n)_{n\in\mathbb{N}}$ generated by the algorithm remains bounded, the sequence converges to a limit point x^*, where

$$x^* = \arg\min_{x\in\mathbb{R}^d} \left\{ F(x) + \mathbb{E}_{\epsilon\sim P_0} \left[\|D_t(x) - x\|^2 \right] \right\},$$

and $D_t(x)$ is the denoising operator at time t. The expectation is taken over the latent noise ϵ.

This reformulation of Proposition 4 provides a convergence result in the context of PnP denoising, where the algorithm converges to a point that minimizes both the data-fidelity term $F(x)$ and the denoising error with respect to the latent noise ϵ. The introduction of the expectation over ϵ aligns the convergence result with the PnP framework, where the denoising process is averaged over the latent distribution.

7 Experiments and results

Using the provided checkpoint, we conducted tests on mupltiple images on two image disturbance of random image masking with 70% of coverage, and image in-painting. The result can be seen in figure 3. Interestingly the algorithm takes different paths for the same image but for different restoration tasks. We also included a non-face image and while the result for in-painting task is not very good but for the random mask it has restored the image more precisely.

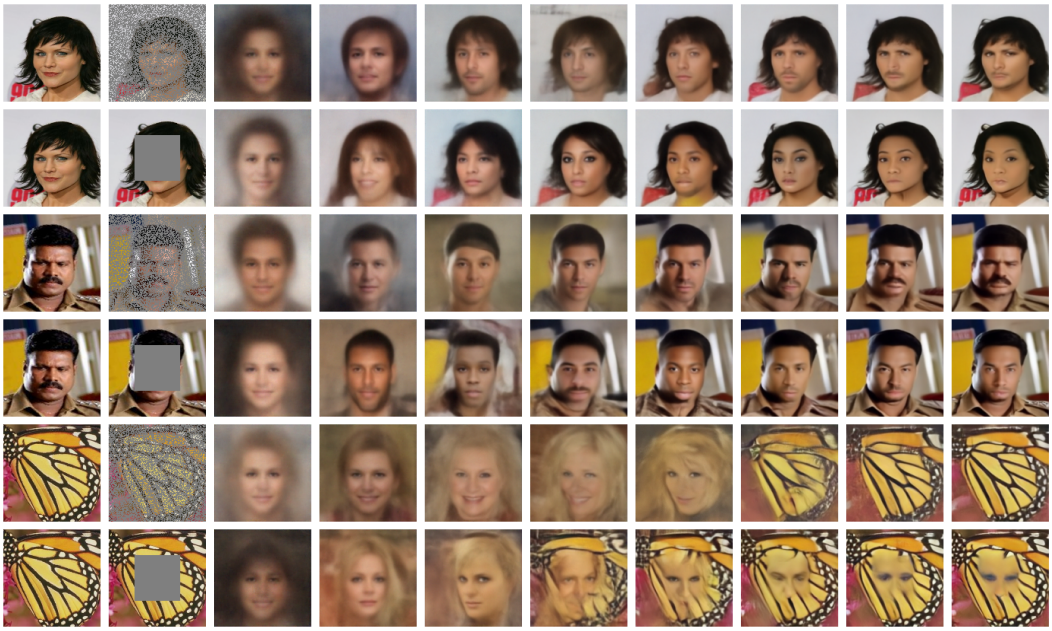

Figure 3: PnP-FM algorithm on multiple images with two different restoration tasks. 1st left: true data; 2nd left: disturbed data.

Figure 4: PnP-FM algorithm on image with size of 512×512 with two different restoration tasks. 1st left: true data; 2nd left: disturbed data.

Furthermore, we conduct experiments using larger images. While, the model is mainly trained on 128×128 we test with 4 times bigger images. The results can be seen in figure 4, the model can reconstruct with random mask corruption task but with in-painting task the model fails to restore the image.

References

Kornilov, N., Mokrov, P., Gasnikov, A., and Korotin, A. (2024). Optimal flow matching: Learning straight trajectories in just one step.

Lee, S., Kim, B., and Ye, J. C. (2023). Minimizing trajectory curvature of ode-based generative models.

Martin, S., Gagneux, A., Hagemann, P., and Steidl, G. (2025). Pnp-flow: Plug-and-play image restoration with flow matching.

Tong, A., Fatras, K., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Wolf, G., and Bengio, Y. (2023). Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*.

Yang, L., Zhang, Z., Zhang, Z., Liu, X., Xu, M., Zhang, W., Meng, C., Ermon, S., and Cui, B. (2024). Consistency flow matching: Defining straight flows with velocity consistency.