SYNTHETIC LEARNING FOR IMAGE DENOISING: TWO SIDES OF THE SAME COIN

IMAGES Team, Télécom Paris

Raphaël Achddou

December 9th,2025

A3SI, LIGM, ESIEE Paris

https://rachddou.github.io
raphael.achddou@esiee.fr

DEEP IMAGE DENOISING

Noise in Imaging

In all imaging fields, images are corrupted with noise:

$$y = x + n,\ n \sim \mathrm{p}_{\mathrm{noise}} \tag{1}$$

- y: the noisy observation
- x: the ground truth

Noise in Imaging

In all imaging fields, images are corrupted with noise:

$$y = x + n,\ n \sim \mathrm{p}_{\mathrm{noise}} \tag{1}$$

- y: the noisy observation
- x: the ground truth

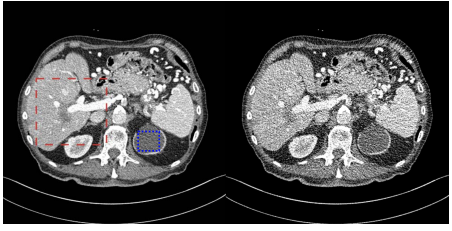


Figure: CT scan with low dose

In all imaging fields, images are corrupted with noise:

$$y = x + n,\ n \sim \mathrm{p}_{\mathrm{noise}} \tag{1}$$

- y: the noisy observation
- x: the ground truth

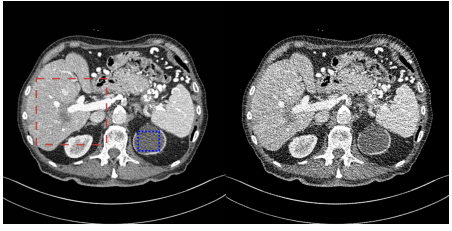


Figure: CT scan with low dose

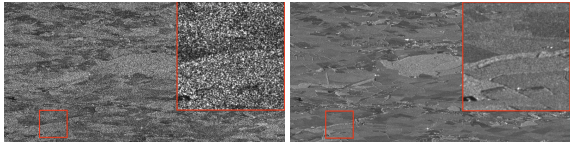


Figure: Speckle Noise in SAR imaging

Noise in Imaging

In all imaging fields, images are corrupted with noise:

$$y = x + n,\ n \sim \mathrm{P}_{\mathrm{noise}} \tag{1}$$

- y: the noisy observation
- x: the ground truth

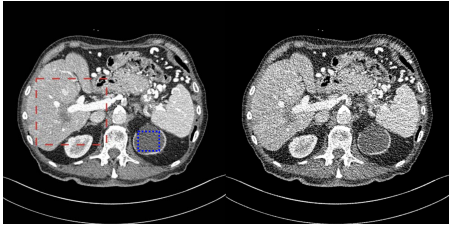


Figure: CT scan with low dose

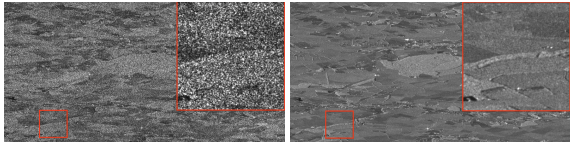


Figure: Speckle Noise in SAR imaging

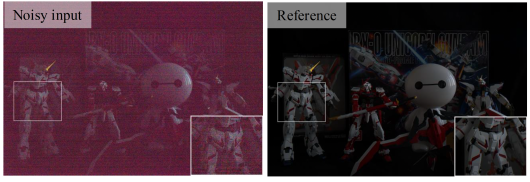


Figure: Camera noise in low-light

For all these cases: difficult / impossible acquisition of noiseless images

$\Longrightarrow$ **Develop denoising algorithms!**

Goal

Given any observation $y = x + n,\ n \sim \mathrm{p}_{\text{noise}}$, where $x \sim \mathrm{p}_{\text{data}}$, find an estimate $\hat{x}(y)$ as close as possible to x.

For all these cases: difficult / impossible acquisition of noiseless images
$\Longrightarrow$ **Develop denoising algorithms!**

Goal

Given any observation $y = x + n,\ n \sim \mathrm{p}_{\text{noise}}$, where $x \sim \mathrm{p}_{\text{data}}$, find an estimate $\hat{x}(y)$ as close as possible to x.

Characterization. The Mean Squared Error:

$$MSE(f) = \mathbb{E}_{x \sim \mathrm{p}_{\text{data}}} ||x - f(x+n)||_2^2 \tag{2}$$

Optimal Estimator. The Minimal Mean Squared Error estimator:

$$f_{MMSE}(y) := \operatorname{argmin}_{f \in F} \Big[\mathbb{E}_{x \sim \mathrm{p}_{\text{data}}} \left(||f(y) - x||^2 \right) \Big] = \mathrm{E}(x|y). \tag{3}$$

What about deep denoisers?

Deep Learning setting

Optimization problem

$$\min_{\theta \in \Theta} \mathbb{E}_{x \sim \hat{\mathrm{p}}_{\mathrm{data}}, n \sim \hat{\mathrm{p}}_{\mathrm{noise}}} ||x + f_\theta(x+n)||_2^2 \tag{4}$$

Solved with a variant of the SGD. $f_{\theta^*} \neq f_{MMSE}$ because of some approximations.

Deep Learning setting

Optimization problem

$$\min_{\theta \in \Theta} \mathbb{E}_{x \sim \hat{\mathrm{p}}_{\mathrm{data}}, n \sim \hat{\mathrm{p}}_{\mathrm{noise}}} ||x + f_\theta(x+n)||_2^2 \tag{4}$$

Solved with a variant of the SGD. $f_{\theta^*} \neq f_{MMSE}$ because of some approximations.

The architecture. A restricted space of possible solutions:

$$\hat{F} = \{f_\theta | \theta \in \Theta\} \tag{5}$$

$$f_\theta(x) := \sigma_N(W_N^T(\dots(\sigma_1(W_1^T x + b_1))\dots) + b_N) \tag{6}$$

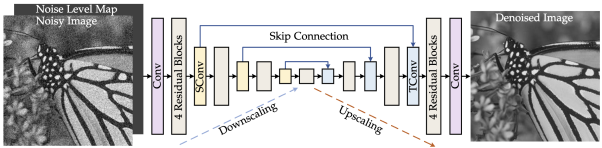


Figure: DRUNet architecture [Zhang et al., 2021]

Deep Learning setting

Optimization problem

$$\min_{\theta \in \Theta} \mathbb{E}_{x \sim \hat{\mathrm{p}}_{\text{data}}, n \sim \hat{\mathrm{p}}_{\text{noise}}} ||x + f_\theta(x+n)||_2^2 \tag{4}$$

Solved with a variant of the SGD. $f_{\theta^*} \neq f_{MMSE}$ because of some approximations.

The dataset. A discrete approximation of p_{data}, given a dataset D:

$$\hat{\mathrm{p}}_{\text{data}} = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_i}, \{x_i\}_{i \leq N} = \mathrm{D} \tag{5}$$

Figure: Examples from the Imagenet dataset

Deep Learning setting

Optimization problem

$$\min_{\theta \in \Theta} \mathbb{E}_{x \sim \hat{\mathrm{p}}_{\text{data}}, n \sim \hat{\mathrm{p}}_{\text{noise}}} ||x + f_\theta(x+n)||_2^2 \tag{4}$$

Solved with a variant of the SGD. $f_{\theta^*} \neq f_{MMSE}$ because of some approximations.

The noise. A Gaussian approximation of noise

$$\hat{\mathrm{p}}_{\text{noise}} = \mathrm{N}(0, \sigma^2 \mathrm{Id}) \tag{5}$$

- easy sampling
- iid hypothesis

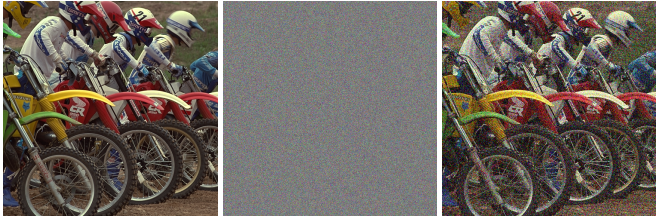


Figure: $x \sim \hat{\mathrm{p}}_{\text{data}}$, $n \sim \hat{\mathrm{p}}_{\text{noise}}$, $y = x + n$

Nonetheless! Extremely good performance of modern deep denoisers.

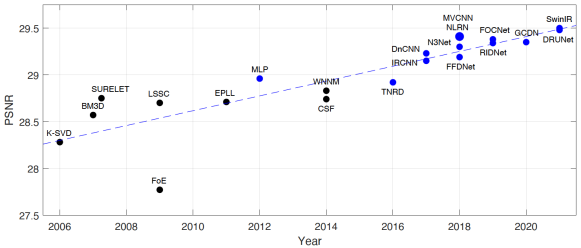


Figure: Evolution of Denoising performances over time

Figure: Image denoising

Nonetheless! Extremely good performance of modern deep denoisers.

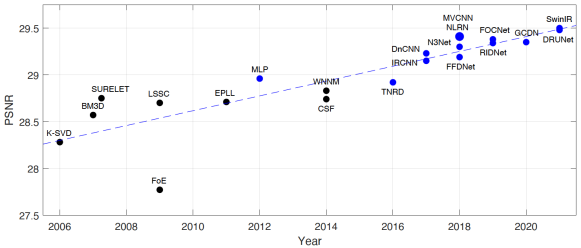


Figure: Evolution of Denoising performances over time

Figure: Image denoising

Nonetheless! Extremely good performance of modern deep denoisers.

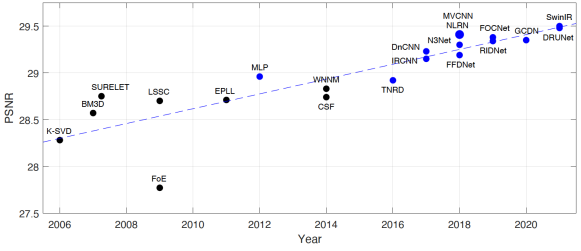


Figure: Evolution of Denoising performances over time

Figure: : NL Means

Nonetheless! Extremely good performance of modern deep denoisers.

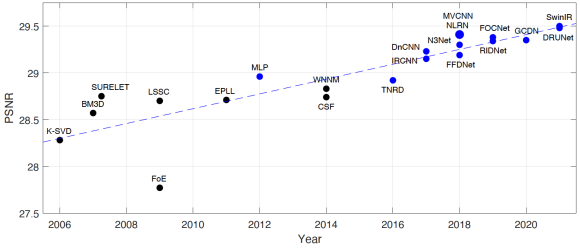


Figure: Evolution of Denoising performances over time

Figure: : BM3D

Nonetheless! Extremely good performance of modern deep denoisers.

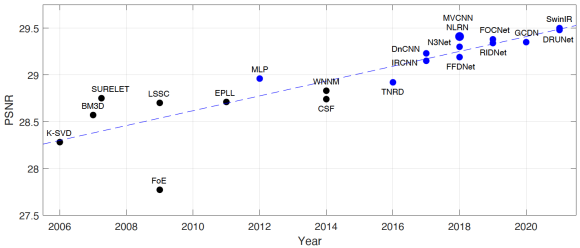


Figure: Evolution of Denoising performances over time

Figure: : DRUNet

BUT!

These approximations $\Longrightarrow$ some limitations for deep denoisers.

These approximations $\implies$ some limitations for deep denoisers.

$$\hat{p}_{\text{data}} \neq p_{\text{data}}$$

- datasets are finite
- datasets are therefore imbalanced $\implies$ **biases in the learnt denoisers**
- **Privacy / safety:** real images contain sensitive information

BUT!

These approximations $\Longrightarrow$ some limitations for deep denoisers.

$\hat{p}_{\text{data}} \neq p_{\text{data}}$

- datasets are finite
- datasets are therefore imbalanced $\Longrightarrow$ **biases in the learnt denoisers**
- **Privacy / safety:** real images contain sensitive information

$\hat{p}_{\text{noise}} \neq p_{\text{noise}}$

- real noise is much more complex than Gaussian noise
- not iid, signal dependent, correlated.
- $\Longrightarrow$ **sub-optimal performance on real-world scenarii**

Other related issues: Surface-level explainability, unpredictible hallucinations, lack of generalization.

A common strategy: Synthetic Learning

Definition

Training machine learning models on artificially generated data rather than real-world data.

A common strategy: Synthetic Learning

Definition

Training machine learning models on artificially generated data rather than real-world data.

part I

$$\hat{\mathrm{P}}_{\mathrm{data}} \rightarrow \mathrm{P}_{\mathrm{VibrantLeaves}}$$

Vibrant Leaves: a new statistical image model replicating key image properties

- sampling arbitrary amounts
- abstract images $\rightarrow$ no semantic/sensitive information
- property factorization $\rightarrow$ explainability

Definition

Training machine learning models on artificially generated data rather than real-world data.

part I

$$\hat{\mathrm{P}}_{\text{data}} \rightarrow \mathrm{P}_{\text{VibrantLeaves}}$$

Vibrant Leaves: a new statistical image model replicating key image properties

- sampling arbitrary amounts
- abstract images $\rightarrow$ no semantic/sensitive information
- property factorization $\rightarrow$ explainability

part II

$$\mathrm{N}(0, \sigma^2) \rightarrow \text{2 Shots in the Dark}$$

a novel camera noise generator

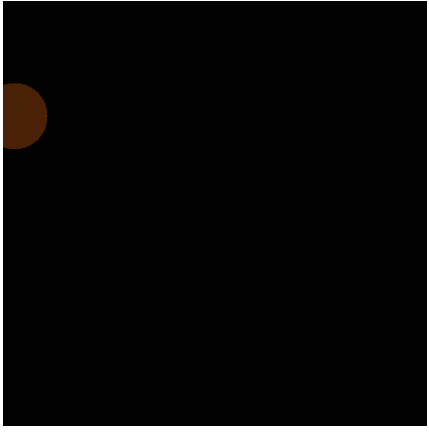
- very little calibration data required
- accurate modeling of noise correlations and statistical properties
- $\Longrightarrow$ SOTA performance on real-world low-light denoising benchmarks

Part I : Replacing $\hat{p}_{\text{data}}$ with $p_{\text{VibrantLeaves}}$

THE DEAD LEAVES MODEL: BASIC CONCEPT

The Dead Leaves Model: Basic concept

A random superimposition of shapes of random size, color, and positions.



The Dead Leaves Model: Mathematical formulation

the dead leaves model

a random process $(x_i, t_i, X_i)_{i \in \mathbb{N}}$

- $x_i, t_i \sim \mathrm{P} = \Sigma \delta_{x_i, t_i}$, a Poisson point process on $\mathbb{R}^2 \times (-\infty, 0]$
- X_i random sets of $\mathbb{R}^2$; usually the set of disks of radius $r_i \sim p(r)$

The Dead Leaves Model: Mathematical formulation

the dead leaves model

a random process $(x_i, t_i, X_i)_{i\in\mathbf{N}}$

- $x_i, t_i \sim \mathrm{P} = \Sigma\delta_{x_i,t_i}$, a Poisson point process on $\mathbb{R}^2 \times (-\infty, 0]$
- X_i random sets of $\mathbb{R}^2$; usually the set of disks of radius $r_i \sim p(r)$

Useful definitions:

- **A leaf:** the set of positions $x_i + X_i$
- **The visible part:** the positions of a leaf which are not covered by previous leaves: $V_i = (x_i + X_i) \backslash \bigcup_{t_j\in(t_i,0)} (x_j + X_j)$
- **dead leaves tesselation**: $T = \bigcup_i V_i$
- **the dead leaves image:** the result of coloring the visible parts with $c_i \sim q(c)$

Examples I

- $r \sim const$
- $c \sim \mathrm{U}([0,1]^3)$

Properties

artificial colors, constant shape size
$\rightarrow$ not very natural.

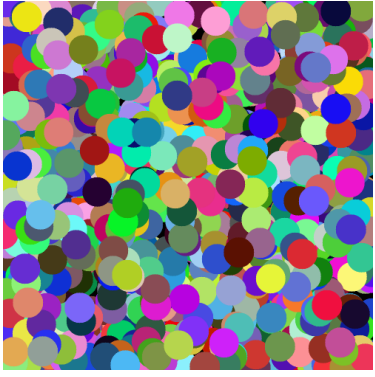


Figure: Process sample

Examples II

- $r \sim const$

Properties

colors sampled from a real image's histogram
→ same color distribution
over-simplistic geometry.

- $c \sim$ color_histo$(I), I$: natural image

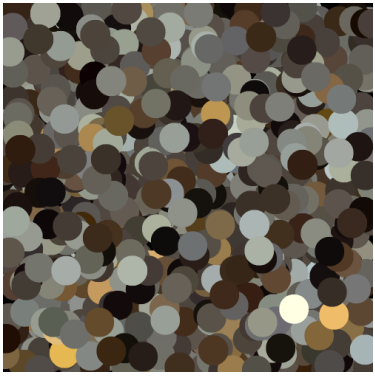


Figure: Process sample

- $r \sim p(r) = C.r^{-\alpha}$, usually $\alpha = 3$

Properties

natural colors + and power law distribution of the radius. **Special case:** $\alpha = 3 \rightarrow$ scale invariance property.

- $c \sim$ color_histo(I), I: natural image

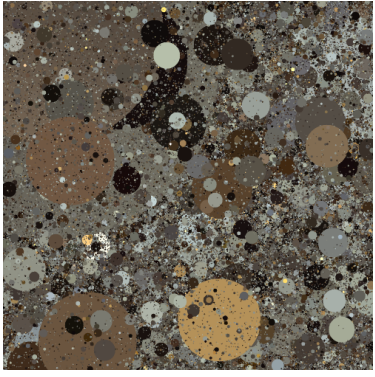
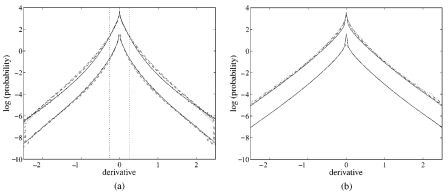


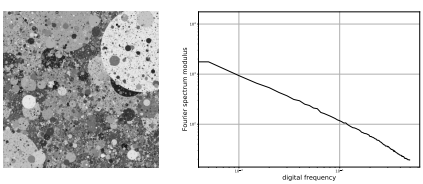
Figure: Process sample

Advantages of the Dead Leaves Model

- few parameters /good compromise between complexity and fidelity
- ”natural” statistical properties.
- direct control over contrast, colors.
- direct control over invariance properties:
  - scale invariance
  - rotation invariance
  - shift invariance
  - contrast invariance



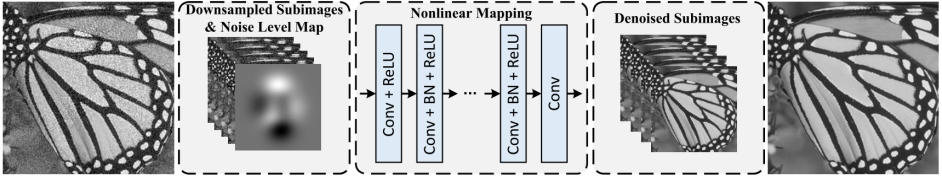
(a) Distribution of the gradient



(b) Average 1-D power spectrum

Case study: Training FFDNet with dead leaves images

FFDNet: a lightweight image denoising CNN [Zhang et al., 2018].
Architecture.

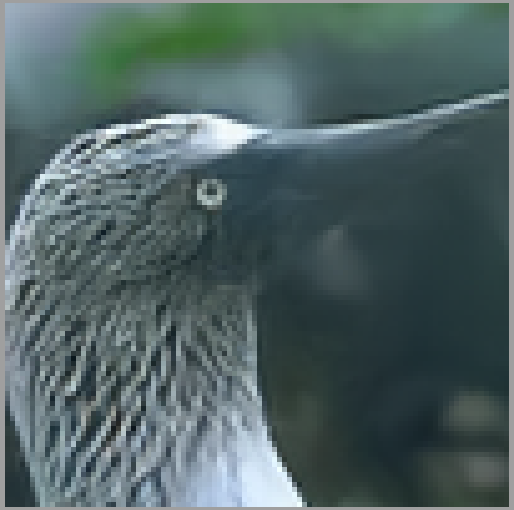


Experimental Protocol, presented in [Achddou et al., 2021] *

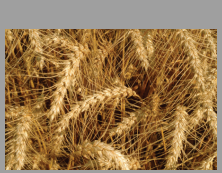
- Generate 10k DL images of size $(500, 500, 3)$ / specific configuration as GT,
- Train FFDNet for color image denoising for each dataset
- Test the models on natural image benchmarks *(CBSD68, Kodak24, McMaster).*

*Synthetic Images as a Regularity Prior for Image Restoration Neural Networks, SSVM 2021

Training with natural images

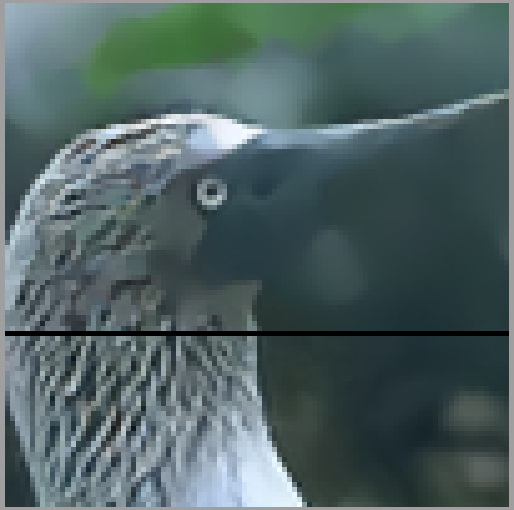


Database: 5000 images de from the Waterloo Exploration Database.



PSNR: 31.54 dB

Training with dead leaves images with a fixed radius



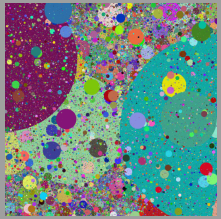
Database: a non gaussian random process of dead leaves images with a fixed radius.

PSNR: 30.1 dB (-1.4)

Training with dead leaves images with random radii

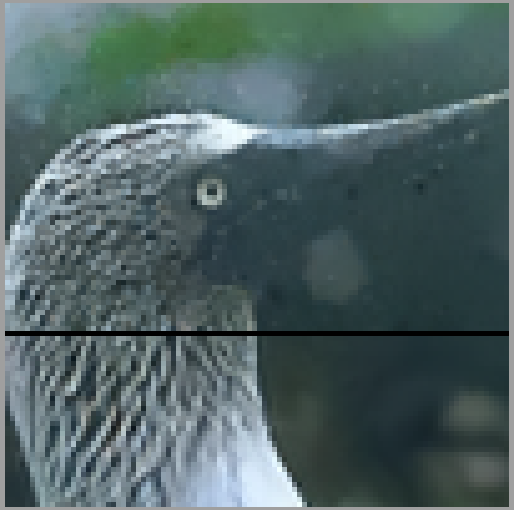
Database:

- DL images with scaling properties (power law of radii with $\alpha = 3$,$r_{\min} = 1$, $r_{\max} = 2000$)
- Colors uniformly drawn in the RGB cube.

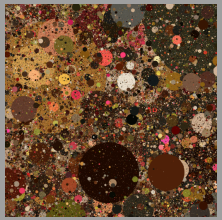


PSNR: 29.6 dB (-1.8)

Training with DL images with natural colors



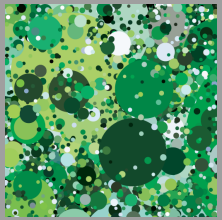
- DL images with scaling properties (power law of radii with $\alpha = 3$,$r_{\min} = 1$, $r_{\max} = 2000$)
- Colors drawn from natural images histograms



PSNR: 30.61 dB (-0.9)

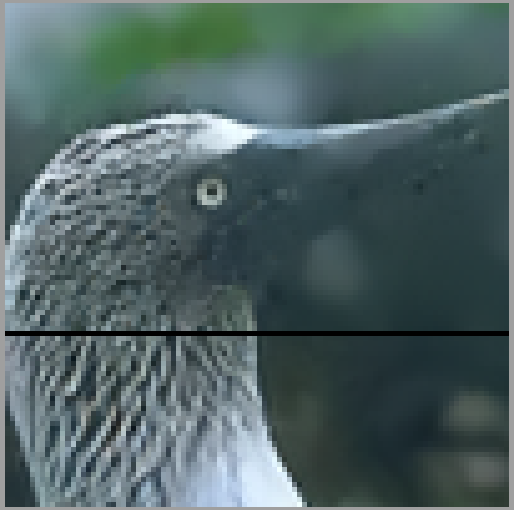
Training with DL images $r_{min} = 16$

- DL images with scaling properties (power law of radii with $\alpha = 3$,$r_{\min} = 16$, $r_{\max} = 2000$)
- Colors drawn from natural images histograms

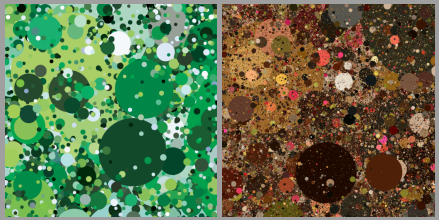


PSNR: 30.55 dB (-1)

Training with a mix of DL images with $r_{min} \in \{1, 16\}$



Database: A mix of DL images with $\alpha = 3, r_{min} \in \{1, 16\}, r_{max} = 2000$.



PSNR: 30.94 dB (-0.6)

Take-away Messages

Take-away Messages

→ Better understanding of required image properties for NN training:

- **Non-gaussianity** of the image model (occlusions/Clear edges)
- **Scale invariance** property
- **Color distribution** close to natural images
- **Diversity** of the training database

Take-away Messages

Take-away Messages

→ Better understanding of required image properties for NN training:

- **Non-gaussianity** of the image model (occlusions/Clear edges)
- **Scale invariance** property
- **Color distribution** close to natural images
- **Diversity** of the training database

Limitations

- oversimplistic geometry
- textures only arise from very small leaves / otherwise homogeneous areas
- no depth modeling except for occlusions
- a substantial performance gap with training on real images

Take-away Messages

Take-away Messages

→ Better understanding of required image properties for NN training:

- **Non-gaussianity** of the image model (occlusions/Clear edges)
- **Scale invariance** property
- **Color distribution** close to natural images
- **Diversity** of the training database

Limitations

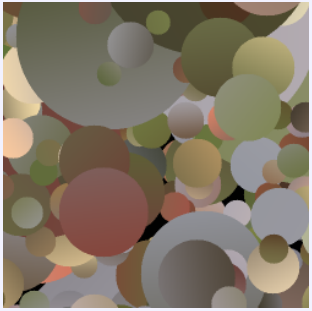
- oversimplistic geometry
- textures only arise from very small leaves / otherwise homogeneous areas
- no depth modeling except for occlusions
- a substantial performance gap with training on real images

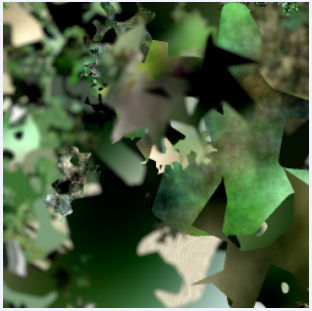
Figure: Comparison of denoising results: natural vs Dead leaves training of DRUNet

THE VIBRANT LEAVES MODEL

Differences between the two models

Properties	Natural Colors	Scaling properties	Depth modelling	Complex Geometry	Repetitive Textures
Dead Leaves model	✓	✓	∼	✗	✗
VibrantLeaves model	✓	✓	✓	✓	✓





GEOMETRY

Figure: Segments of single objects from Pascal-VOC

Geometry: Natural objects vs Dead Leaves model

Figure: Segments of single objects from Pascal-VOC

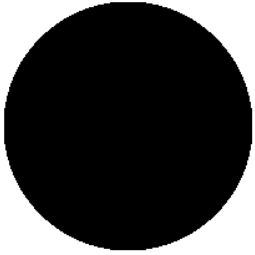


Figure: Geometry of Dead Leaves objects

Observations

- **Dead leaves objects:** disks! only convex shapes with constant curvature

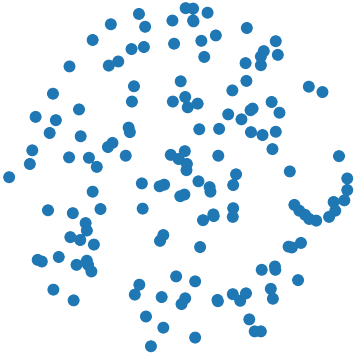


Figure: random points

Shape Generation algorithm

1. Sample N points uniformly in a disk D of radius R (maintain rotation invariance)

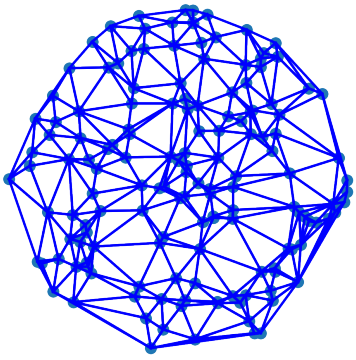


Figure: Delaunay triangles

Shape Generation algorithm

1. Sample N points uniformly in a disk D of radius R (maintain rotation invariance)
2. Compute Delaunay triangulation of the points

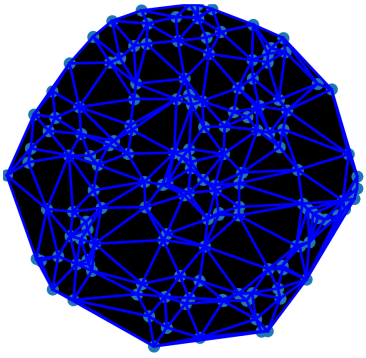


Figure: Concave Hull $\alpha = 0.8$

Shape Generation algorithm

1. Sample N points uniformly in a disk D of radius R (maintain rotation invariance)
2. Compute Delaunay triangulation of the points
3. **Concave Hulls**: remove triangles whose circumradius is larger than $R.\alpha$, without breaking connectivity

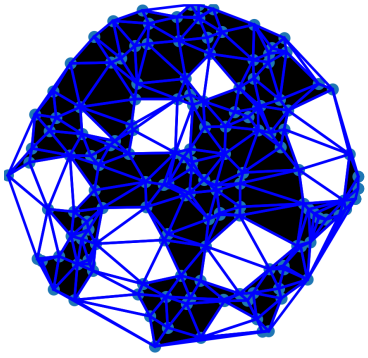


Figure: Concave Hull $\alpha = 0.5$

Shape Generation algorithm

1. Sample N points uniformly in a disk D of radius R (maintain rotation invariance)
2. Compute Delaunay triangulation of the points
3. **Concave Hulls**: remove triangles whose circumradius is larger than $R.\alpha$, without breaking connectivity

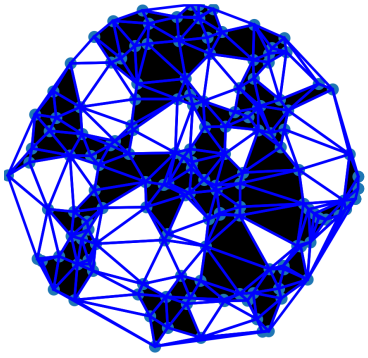


Figure: Concave Hull $\alpha = 0.3$

Shape Generation algorithm

1. Sample N points uniformly in a disk D of radius R (maintain rotation invariance)
2. Compute Delaunay triangulation of the points
3. **Concave Hulls**: remove triangles whose circumradius is larger than $R.\alpha$, without breaking connectivity

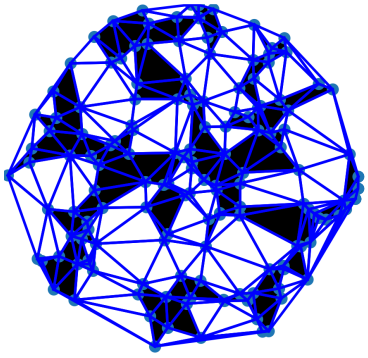


Figure: Concave Hull $\alpha = 0.1$

Shape Generation algorithm

1. Sample N points uniformly in a disk D of radius R (maintain rotation invariance)
2. Compute Delaunay triangulation of the points
3. **Concave Hulls**: remove triangles whose circumradius is larger than $R.\alpha$, without breaking connectivity

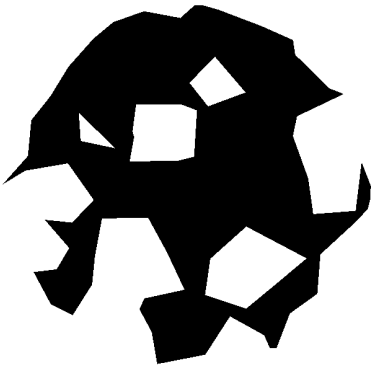


Figure: Original shape

Shape Generation algorithm

1. Sample N points uniformly in a disk D of radius R (maintain rotation invariance)
2. Compute Delaunay triangulation of the points
3. **Concave Hulls**: remove triangles whose circumradius is larger than $R.\alpha$, without breaking connectivity
4. *Curved version*: Convolve with a Gaussian kernel and threshold to obtain smooth shapes

Figure: Smoothened shape

Shape Generation algorithm

1. Sample N points uniformly in a disk D of radius R (maintain rotation invariance)
2. Compute Delaunay triangulation of the points
3. **Concave Hulls**: remove triangles whose circumradius is larger than $R.\alpha$, without breaking connectivity
4. *Curved version*: Convolve with a Gaussian kernel and threshold to obtain smooth shapes

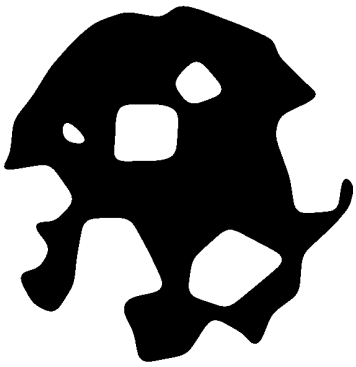


Figure: Binarized shape

Shape Generation algorithm

1. Sample N points uniformly in a disk D of radius R (maintain rotation invariance)
2. Compute Delaunay triangulation of the points
3. **Concave Hulls**: remove triangles whose circumradius is larger than $R.\alpha$, without breaking connectivity
4. *Curved version*: Convolve with a Gaussian kernel and threshold to obtain smooth shapes

Natural vs VibrantLeaves objects

Figure: Segments of single objects from Pascal-VOC

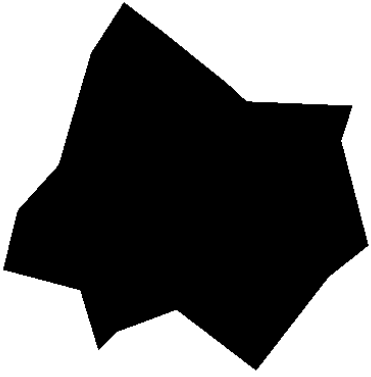


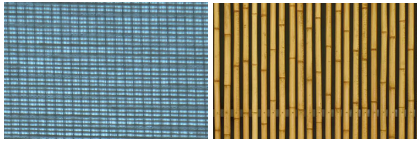
Figure: Objects from the VibrantLeaves shape generator

TEXTURE

What is a texture?

No clear definition of a texture: a pattern that repeats itself with slight modifications at various scales. $\sim$ 2 types of textures

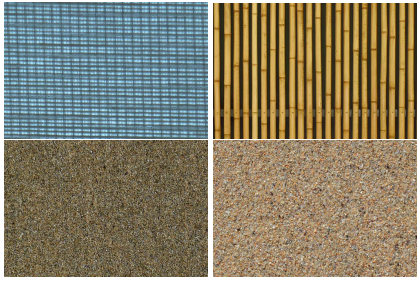
- pseudo-periodic textures



What is a texture?

No clear definition of a texture: a pattern that repeats itself with slight modifications at various scales. $\sim$ 2 types of textures

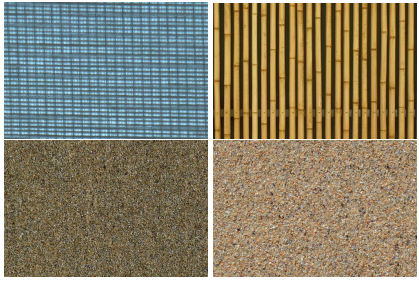
- pseudo-periodic textures
- micro-textures $\sim$ random noise



What is a texture?

No clear definition of a texture: a pattern that repeats itself with slight modifications at various scales. $\sim$ 2 types of textures

- pseudo-periodic textures
- micro-textures $\sim$ random noise

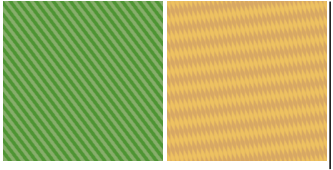


Texture synthesis: a widely research topic, but most methods are **exemplar-based**.
Goal: Propose an **exemplar-free texture generator**, based on principles of *randomness* and *repetitions*.

Pseudo-Periodic Pattern Generator

A model that creates pseudo-periodic interpolation maps T between two colors, with increasing complexity.

1. $T_1(x, \omega) = \sin(\omega x)$ a 1 or 2D sinusoidal map of random period,
2. $T_2(x, \omega) = \text{sigmoid}(\sin(\omega x), \alpha)$, sharper transitions with a logit funtion

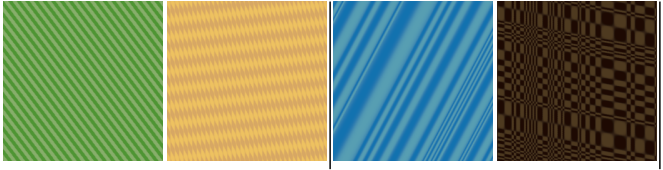
T_2 : Sinusoidal textures	T_3 : Random periods	T_4 : Warped textures
		

Pseudo-Periodic Pattern Generator

A model that creates pseudo-periodic interpolation maps T between two colors, with increasing complexity.

1. $T_1(x, \omega) = \mathbf{sin}(\omega x)$ a 1 or 2D sinusoidal map of random period,
2. $T_2(x, \omega) = \text{sigmoid}(\mathbf{sin}(\omega x), \alpha)$, sharper transitions with a logit funtion
3. $T_3(x) = \text{stack}(\{T_2(\omega_i)\}_{i<n})(x)$ a random oscillatory field

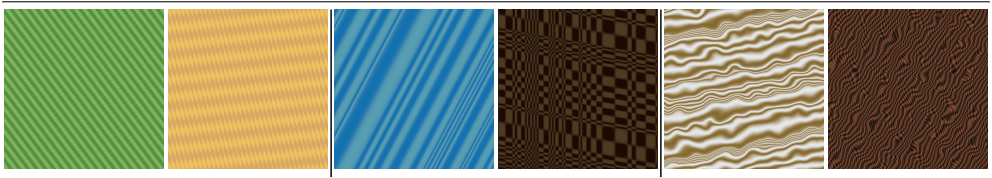
T_2 : Sinusoidal textures	T_3 : Random periods	T_4 : Warped textures



Pseudo-Periodic Pattern Generator

A model that creates pseudo-periodic interpolation maps T between two colors, with increasing complexity.

1. $T_1(x, \omega) = \sin(\omega x)$ a 1 or 2D sinusoidal map of random period,
2. $T_2(x, \omega) = \text{sigmoid}(\sin(\omega x), \alpha)$, sharper transitions with a logit funtion
3. $T_3(x) = \text{stack}(\{T_2(\omega_i)\}_{i<n})(x)$ a random oscillatory field
4. $T_4(x) = \text{warp}(T_3(x))$, a displacement map obtained by filtering noise maps.



Textures modeling: Micro-textures

Common prior of natural images: $|\mathrm{F}(x,\nu)| \simeq \frac{C}{\nu^{\alpha}}$ with $\alpha \in [2-\epsilon, 2+\epsilon]$.

Common prior of natural images: $|\mathrm{F}(x,\nu)| \simeq \frac{C}{\nu^\alpha}$ with $\alpha \in [2-\epsilon, 2+\epsilon]$.

Micro-texture generator

Inspired by the phase randomization texture model of [Galerne et al., 2010]:

1. Generate a **white noise sample** with a natural color histogram:
2. Fix the power spectrum to a power function (isotropic)
3. Reconstruct the image by inverse Fourier transform

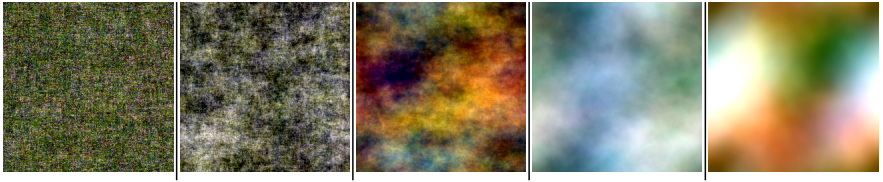
Common prior of natural images: $|\mathrm{F}(x,\nu)| \simeq \frac{C}{\nu^\alpha}$ with $\alpha \in [2-\epsilon, 2+\epsilon]$.

Micro-texture generator

Inspired by the phase randomization texture model of [Galerne et al., 2010]:

1. Generate a **white noise sample** with a natural color histogram:
2. Fix the power spectrum to a power function (isotropic)
3. Reconstruct the image by inverse Fourier transform

$\alpha = 0.5$	$\alpha = 1.0$	$\alpha = 1.5$	$\alpha = 2.0$	$\alpha = 2.5$



DEPTH

Modeling depth and the acquisition process

Depth in natural images

- **Perspective:** a non-linear mapping of 3D to 2D → vanishing points and parralel ines
- **Depth-of-field:** local, non-uniform blur based on object depth
- **Occlusions:** Objects occlude each other in the scene

Depth in the DL model

- only occlusions
- limitted model for physical depth

VL additions: A depth-of-filed simulator and a perspective model for texture maps

Depth-of-field approximation: a tri-plane division of space: blurred background / focused middle ground / blurred foreground

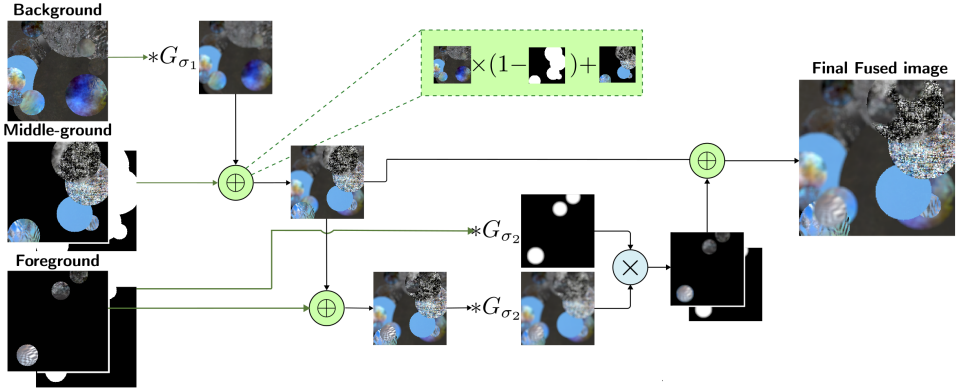


Figure: Diagram of the depth-of-field algorithm. After generating three DL stack (background, middle-ground and foreground), we fuse them by applying blur kernels $G_{\sigma_1}, G_{\sigma_2}$ respectively to the background and foreground.

VibrantLeaves: Examples of images

Figure: Examples of samples from the VibrantLeaves model, which integrates modeling for *geometry, textures, and depth.*

VibrantLeaves: Examples of images

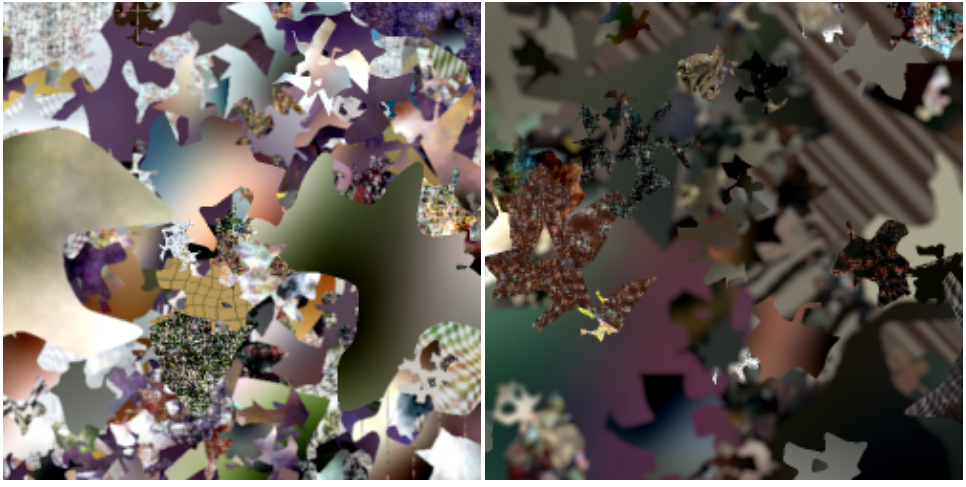


Figure: Examples of samples from the VibrantLeaves model, which integrates modeling for *geometry, textures, and depth.*

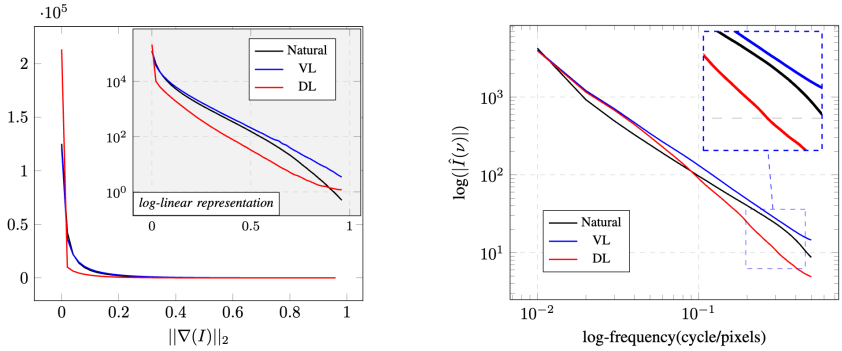
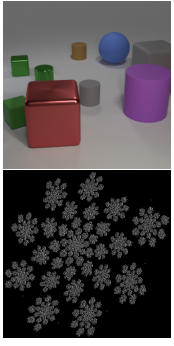


Figure: *Right.* Distribution of the gradient, *Left.* Average 1D power spectrum, for Natural images, DL images, VL images

Vibrant Leaves: FID scores

FID score: a similarity measure at the last hidden layer of a classification NN (InceptionV3). Lower is better / Computed with respect to the traditional training dataset (WaterlooDB).



Metric	*FID* ↓	*KL-Gradient*↓	α_{Spectrum} (R^2) $(_{Nat} = 1.43)$
DL [Achddou et al., 2021]	318	0.286	1.73 *(0.992)*
CleVR [Johnson et al., 2017]	217	0.517	1.67 *(0.992)*
FractalDB [Kataoka et al., 2020]	342	1.91	0.51 *(0.584)*
DL-textured [Baradad et al., 2021]	312	0.228	0.99 *(0.98)*
VL	193	**0.006**	**1.41** *(0.995)*
GTA-5 [Richter et al., 2016]	**186**	0.015	1.49 *(0.982)*

IMAGE RESTORATION RESULTS

Image denoising results

Protocol:

- generate 10K images for every configuration
- train a DRUNet denoiser on these individual datasets
- test on natural image datasets

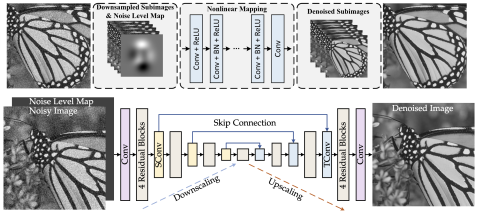
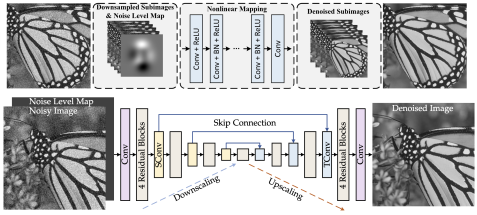


Image denoising results

Protocol:

- generate 10K images for every configuration
- train a DRUNet denoiser on these individual datasets
- test on natural image datasets



	Kodak24		CBSD68		McMaster		Urban100		Bokeh		Average	
Method / σ	25	50	25	50	25	50	25	50	25	50	25	50
Input PSNR	*20.43*	*14.86*	*20.53*	*15.01*	*20.91*	*15.38*	*20.63*	*15.11*	*20.46*	*14.95*	*20.59*	*15.06*
DRUNet Fractal [Kataoka et al., 2020]	17.32	17.06	17.05	16.66	15.57	15.19	16.20	15.93	19.49	18.47	17.13	16.66
DRUNet ClevR [Johnson et al., 2017]	30.42	27.71	29.45	26.64	30.98	28.15	29.43	26.05	37.13	33.91	31.48	28.49
DRUNet DL [Achddou et al., 2021]	30.95	28.09	30.20	27.18	31.25	28.32	29.43	26.05	36.66	33.76	31.69	28.68
DRUNet DLText [Baradad et al., 2021]	31.14	28.11	30.35	27.18	31.33	28.31	29.26	25.81	37.19	33.90	31.85	28.66
DRUNet GTAV [Richter et al., 2016]	32.14	29.20	31.14	28.06	32.43	29.47	31.17	27.90	38.59	35.71	33.09	30.07
DRUNet VL	32.16	29.16	31.21	28.06	32.63	29.59	31.27	27.94	38.70	35.78	33.19	30.11
DRUNet Nat	**32.89**	**29.86**	**31.69**	**28.51**	**33.14**	**30.08**	**32.60**	**29.60**	**39.21**	**36.31**	**33.91**	**30.86**

Table: Image denoising results. Best results are in **bold** and second results are underlined.

Image denoising examples

Figure: Denoising visual results. We compare the same DRUNet architecture trained either on Dead Leaves, Vibrant Leaves or Nat images.

Image denoising examples

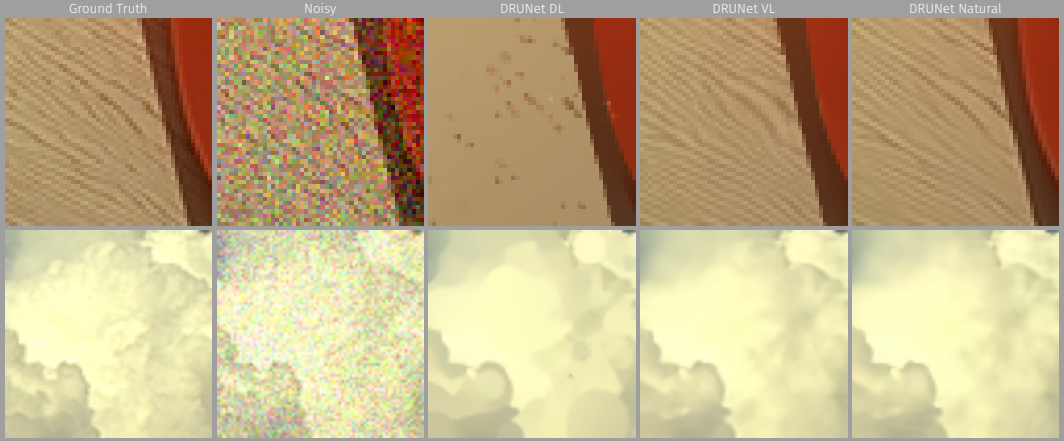


Figure: Denoising visual results. We compare the same DRUNet architecture trained either on Dead Leaves, Vibrant Leaves or Nat images.

BENEFITS OF THE VL MODEL

What are the benefits?

Questions

1. Does the learned denoiser inherit **invariance properties?**
2. Does our image model lead to faster **training convergence?**
3. Can we **isolate image properties** that are crucial for image restoration NNs?
4. Can we verify that the network has learnt these principles?

Dead leaves images are supposed to have many invariances:

- Rotation
- Scale
- Contrast
- Shift

Is our learnt denoiser invariant to these? *or better than the natural baseline...*

Dead leaves images are supposed to have many invariances:

- Rotation **??**
- Scale **??**
- ~~Contrast~~
- ~~Shift~~

Is our learnt denoiser invariant to these? *or better than the natural baseline...*

Dead leaves images are supposed to have many invariances:

- Rotation ??
- Scale ??
- ~~Contrast~~
- ~~Shift~~

Is our learnt denoiser invariant to these? *or better than the natural baseline...*

Testing protocol

- Test both models at a fixed $\sigma = 25$, while varying the distortion level θ.
- Measure the performance difference $\Delta_{\text{model}}(\theta)$ induced by the distortion.
- Report $\xi(\theta) = \Delta_{\text{VL}}(\theta) - \Delta_{\text{Nat}}(\theta) \rightarrow$ if $\xi > 0$, the denoiser is more resilient to the distortion.

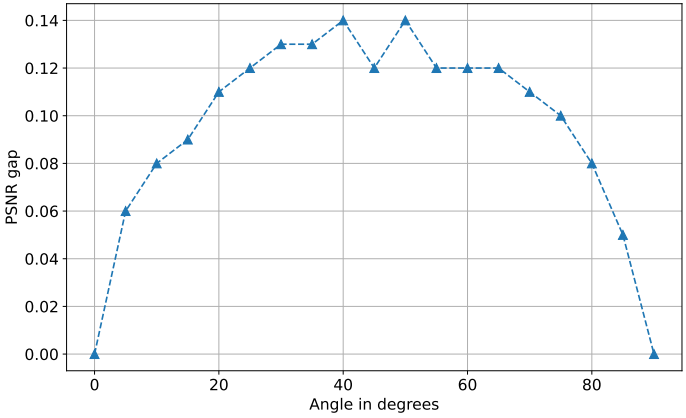


Figure: Performance gap for rotation invariance

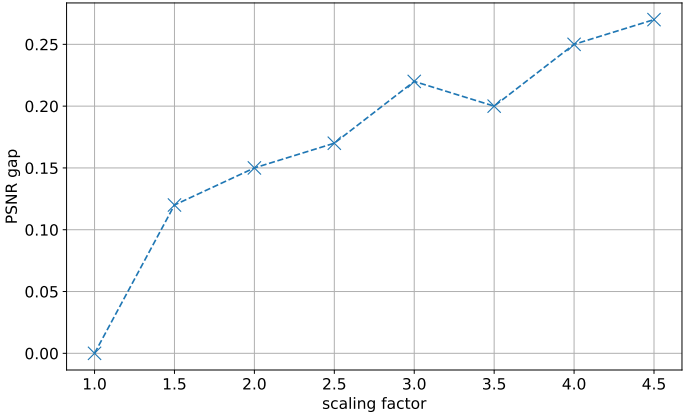


Figure: Performance gap for scale invariance

Goal: Visualize the prior $\tilde{p}_{\mathrm{VibrantLeaves}}$ learned by the denoising network.

- **Turn denoisers into generators!** Diffusion models are denoisers!
- For a denoiser trained on any p_d, use score-based algorithms to sample from the implicitly learnt prior $\tilde{p}_d$ [Kadkhodaie and Simoncelli, 2021, Leclaire et al., 2025]
- **Not Ideal:** Classic denoisers are not trained on all noise levels.

Goal: Visualize the prior $\tilde{p}_{\mathrm{VibrantLeaves}}$ learned by the denoising network.

- **Turn denoisers into generators!** Diffusion models are denoisers!
- For a denoiser trained on any p_d, use score-based algorithms to sample from the implicitly learnt prior $\tilde{p}_d$ [Kadkhodaie and Simoncelli, 2021, Leclaire et al., 2025]
- **Not Ideal:** Classic denoisers are not trained on all noise levels.

Background on diffusion models

- $p_{source} \sim \mathrm{N}(0, \mathrm{Id}), p_{target} \sim \mathrm{p_{data}}$
- Forward Process: $p_t = p_{t-1} * \mathrm{N}(0, \sigma_t.\mathrm{Id})$, such that $p_0 = \mathrm{p_{data}}$ and $p_T \simeq p_{source}$
- Backward process: $x_T \sim p_{source}$ iteratively denoise the image with a denoiser D_θ such that $x_0 \sim \mathrm{p_{data}}$ (with noise injection)
- Tweedie equality: $D_\theta(y, \sigma) - y = \sigma^2 \nabla \log p(y) \rightarrow$ an iterative deblurring of the noisy distribution.

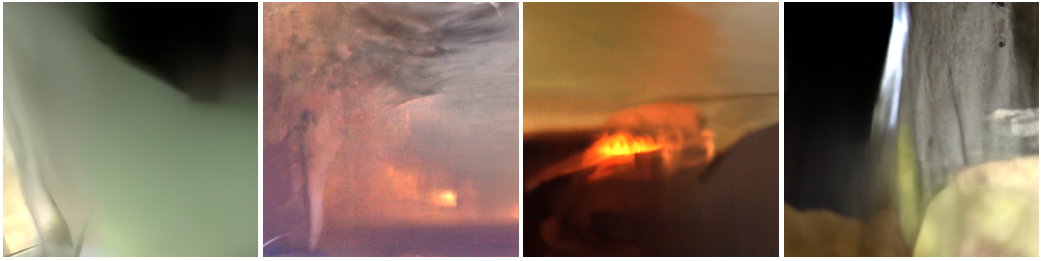


Figure: Images sampled from the prior learnt by a DRUNet denoiser trained on Natural Images.

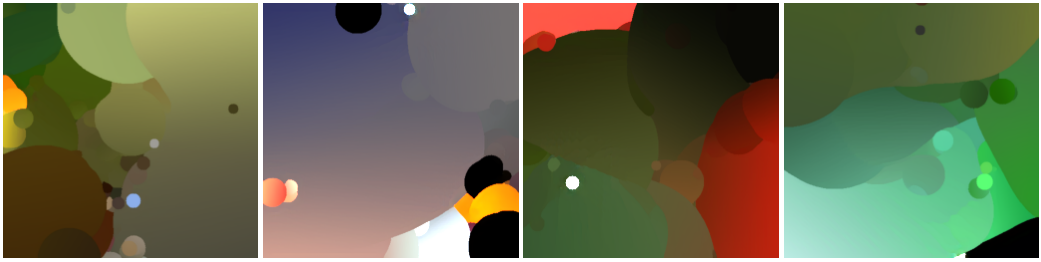


Figure: Images sampled from the prior learnt by a DRUNet denoiser trained on basic Dead Leaves images.

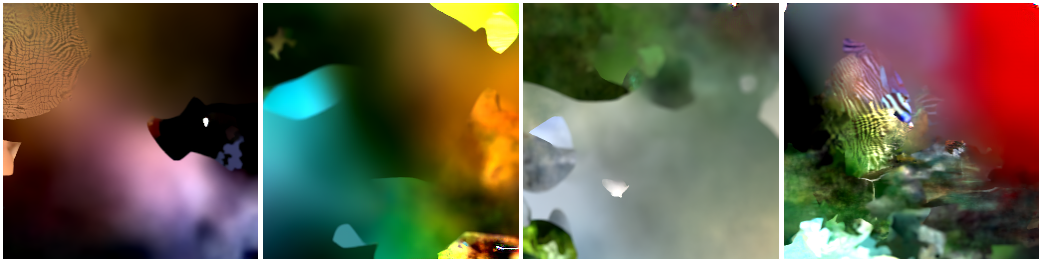


Figure: Images sampled from the prior learnt by a DRUNet denoiser trained on Vibrant Leaves.

Learnt properties from Vibrant Leaves images:

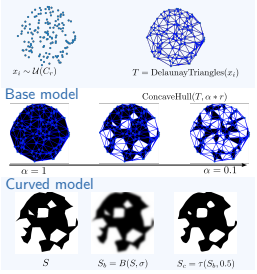
- ✓ occlusions,
- ✓ complex shapes,
- ✓ micro-textures and pseudo-periodic textures,
- ✓ depth-of-field

VibrantLeaves Pseudo-code

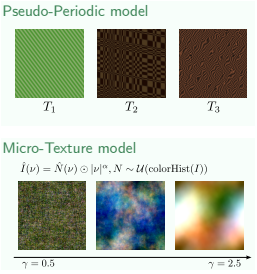
```
define leavesStack(r_min,r_max,α,N):
  img = zeros(h,w,3)
  for (i= 0, i<N, i++):
    r = rdm.power(r_min,r_max,α)
    shape = shapeGenerator(r)
    texture = textureGenerator()
    render_shape = shape*texture
    img=stack(render_shape,img)
Return img

sb,sm,sf = (leavesStack(...,N_i))_{i ≤ 3}
final_img = depthFuse(sb,sm,sf)
```

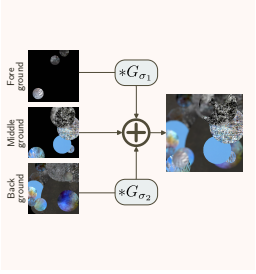
shapeGenerator()



textureGenerator()



depthFuse()



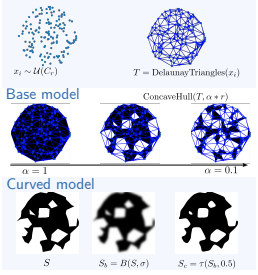
Overview

VibrantLeaves Pseudo-code

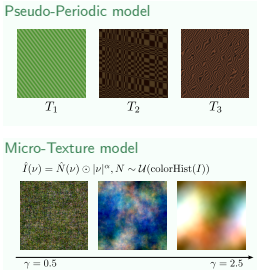
```
define leavesStack(r_min,r_max,α,N):
  img = zeros(h,w,3)
  for (i= 0, i<N, i++):
    r = rdm.power(r_min,r_max,α)
    shape = shapeGenerator(r)
    texture = textureGenerator()
    render_shape = shape*texture
    img=stack(render_shape,img)
Return img

sb,sm,sf = (leavesStack(...,N_i))_{i≤3}
final_img = depthFuse(sb,sm,sf)
```

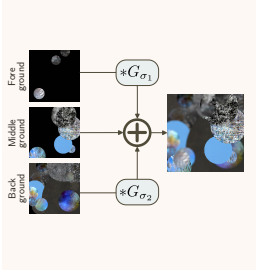
shapeGenerator()



textureGenerator()



depthFuse()

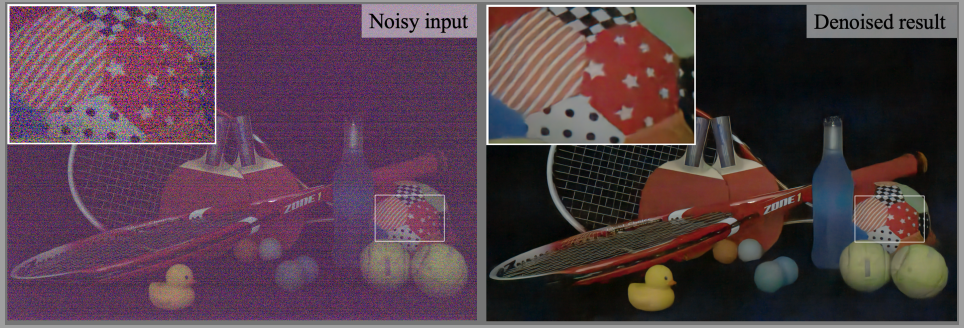


- comparable image denoising restoration with training on $\hat{\text{p}}_{\text{data}}$
- a model based only on geometry and textures
- robustness to some distortions
- a more interpretable model (prior sampling / ablation studies)

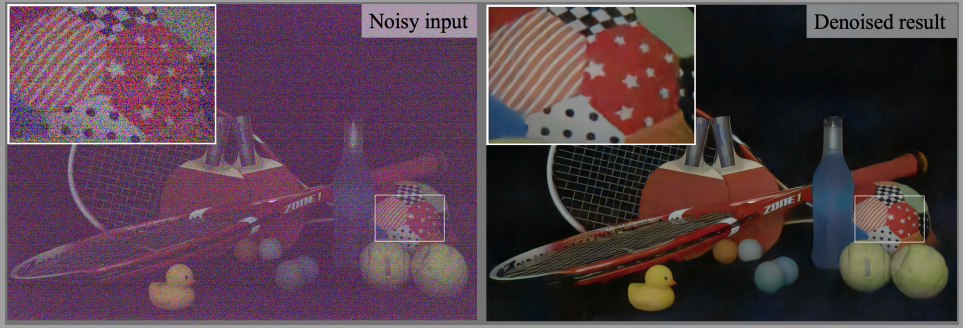
Part II : p_{noise} 2-Shots in the Dark

REAL-WORLD IMAGE DENOISING

Noise Examples



Noise Examples



Real-world RAW noise is far from Gaussian! $p_{\text{noise}} \neq \mathrm{N}(0, \sigma^2 \mathrm{Id})$

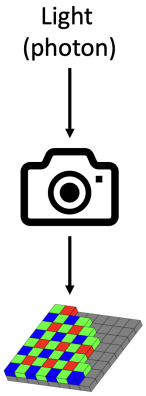


Figure: Photon acquisition with a CMOS sensor

Noise Sources during Image Acquisition:

Signal-dependent Noises:

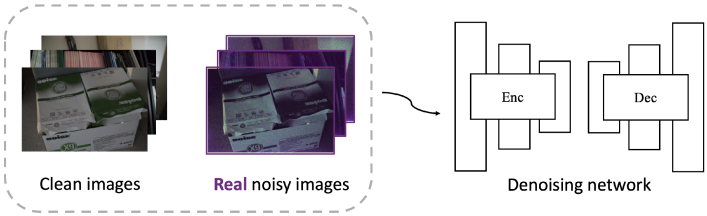
- Photon shot noise
- Photon response Non Uniformity (PRNU)

Signal-independent Noises:

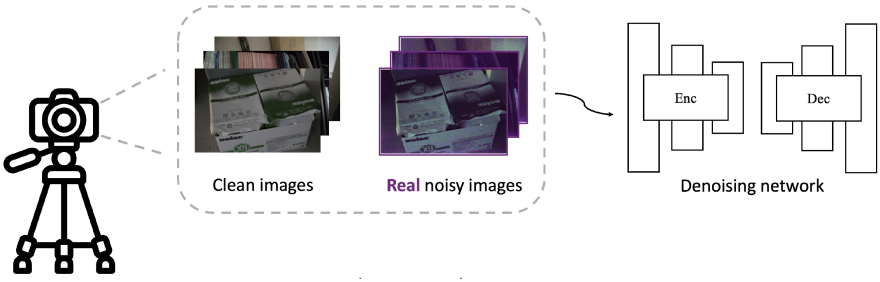
- Read-out noise
- Quantization noise
- Banding pattern noise
- Fixed Pattern Noise (FPN)
- ...

$$Y = N_{\text{shot}}(X, \text{ISO}) + N_{\text{independent}}(\text{ISO}) \tag{6}$$

Training on real paired datasets:



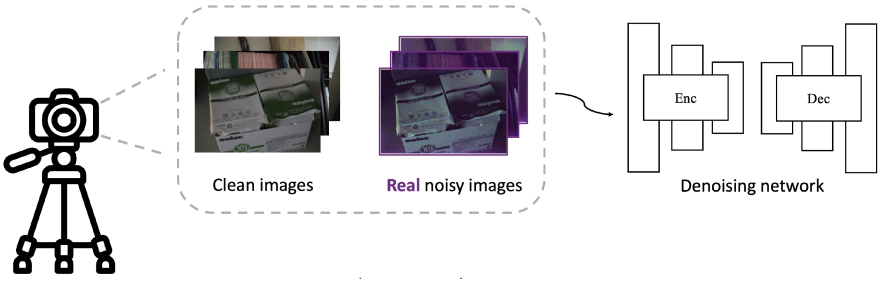
Training on real paired datasets:



Dataset Acquisition:

- Camera mounted on a tripod $\rightarrow$ $\sim$ no movement and alignement
- Only static scenes

Training on real paired datasets:



Limitations:

- **Cumbersome:** time consuming
- **Limited diversity:** limited number of samples + only static scenes in controlled environments
- **Not Generalizable:** camera specific datasets.

Alternative: Synthetic paired data

Original training set: $D = \{x_i, y_i\}_{i \leq N_{pairs}} \cup \{z_j\}_{j \leq N_{calibration}}$

Alternative: Synthetic paired data

Original training set: $D = \{x_i, y_i\}_{i \leq N_{pairs}} \cup \{z_j\}_{j \leq N_{calibration}}$

Alternative: Synthetize noisy data with a realistic noise model!

Alternative: Synthetic paired data

Original training set: $D = \{x_i, y_i\}_{i \leq N_{pairs}} \cup \{z_j\}_{j \leq N_{calibration}}$

Alternative: Synthetize noisy data with a realistic noise model!

Advantages

- arbitrary amounts of noisy samples
- Not limited to scenes x_i from D
- less labor intensive

Alternative: Synthetic paired data

Original training set: $D = \{x_i, y_i\}_{i \leq N_{pairs}} \cup \{z_j\}_{j \leq N_{calibration}}$

Alternative: Synthetize noisy data with a realistic noise model!

Advantages

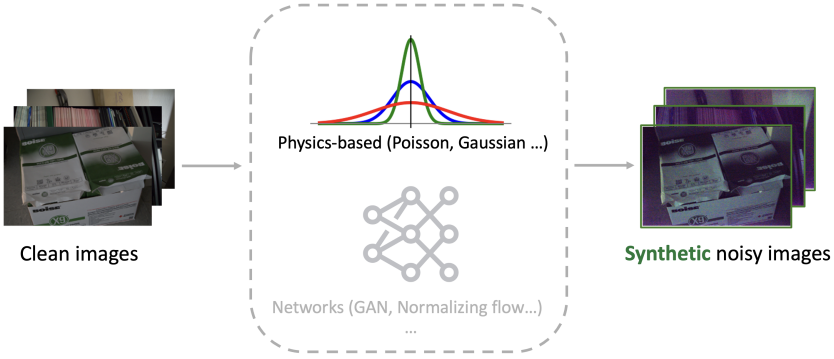
- arbitrary amounts of noisy samples
- Not limited to scenes x_i from D
- less labor intensive

Problem statement:

Given a subset $A \subseteq D$, fit the parameters of a G_θ such that:

$$\theta^\star = \arg\min_{\theta \in \Theta} D(\hat{p}_{\text{noise}} | p_{G_\theta}) \tag{7}$$

Related Works

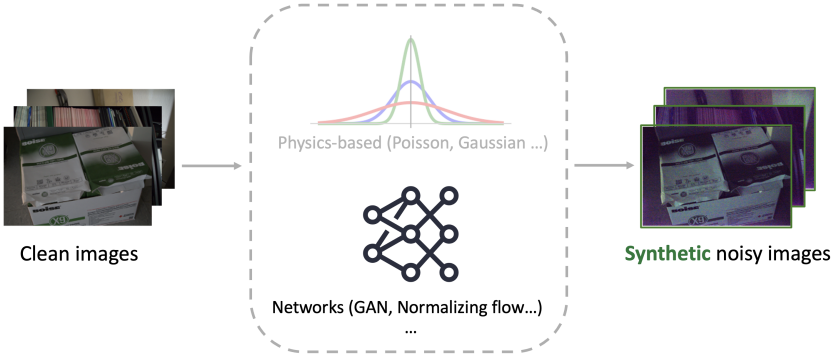


Simple parametric models:

- Gaussian: $n \sim \mathrm{N}(0, \sigma^2)$
- Poisson-Gaussian: $y \sim \mathrm{N}(x, \lambda x^2 + \sigma^2)$
- ELD [Wei et al., 2021]:
  $Y = \frac{X}{\gamma} + N_s\left(\frac{X}{\gamma}\right) + N_r + N_b + N_q,$

- few parameters $\rightarrow$ interpretable
- estimation prone to inaccuracies
- Not very precise modeling

Related Works



Deep Generative models:

- Normalizing Flows: NoiseFlow [Abdelhamed et al., 2019]
- GANs: LRD [Zhang et al., 2023]
- Diffusion Models: NoiseDiff [Lu et al., 2025]

- Not always expressive enough (NF)
- Good noise modeling performance for DM
- Requires large amounts of data

Calibration data: $\{z_i\}_{i \leq N_{calibration}}$

- Noisy images
- *Dark Frames*: images obtained in complete darkness at specific settings

Method	Category	# of real pairs	# of dark frames (per ISO)
LRD	Learning	1865	400
NoiseDiff	Learning	1865	400
PMN	Non-learning	1865	400
ELD	Non-learning	0	Several
Poisson-Gaussian	Non-learning	0	0

Our setting

Calibration data: $\{z_i\}_{i \leq N_{calibration}}$

- Noisy images
- *Dark Frames*: images obtained in complete darkness at specific settings

Method	Category	# of real pairs	# of dark frames (per ISO)
LRD	Learning	1865	400
NoiseDiff	Learning	1865	400
PMN	Non-learning	1865	400
ELD	Non-learning	0	Several
Poisson-Gaussian	Non-learning	0	0
Ours	**Non-learning**	**0**	**1**

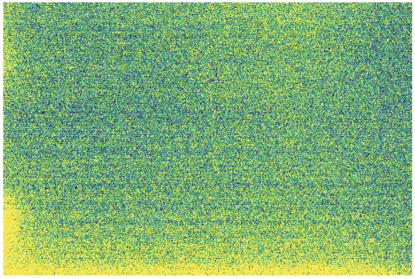
With one Dark Frame and one noisy image per iso $\rightarrow$ find the best G_θ

2-SHOTS IN THE DARK: METHOD

Overview

Noisy image

Dark frame

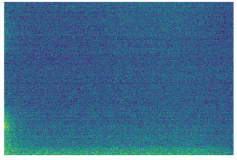


- Noisy image → shot noise estimation
- Dark frame → all signal independent noise sources

Our initial model inspired by texture synthesis

Our methods is largely inspired by **Random Phase Textures (RPN)** $\rightarrow$ a texture synthesis model for **stationary micro-textures**

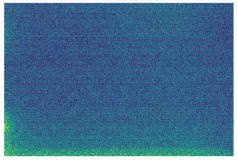
I_{dark}



Our initial model inspired by texture synthesis

Our methods is largely inspired by **Random Phase Textures (RPN)** $\rightarrow$ a texture synthesis model for **stationary micro-textures**

I_{dark}

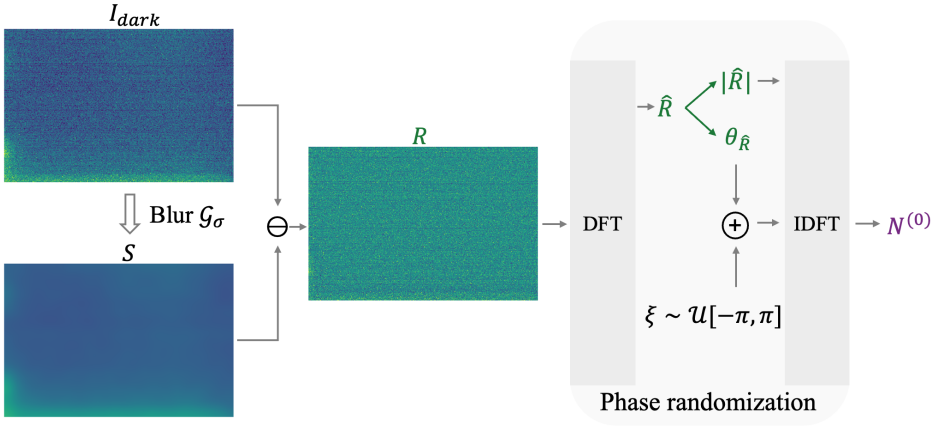


$\Downarrow$ Blur $\mathcal{G}_\sigma$

S

Our initial model inspired by texture synthesis

Our methods is largely inspired by **Random Phase Textures (RPN)** → a texture synthesis model for **stationary micro-textures**



Denoising Results

Testing our model: synthetize noise / add it to real images / Train a denoising Unet with this data

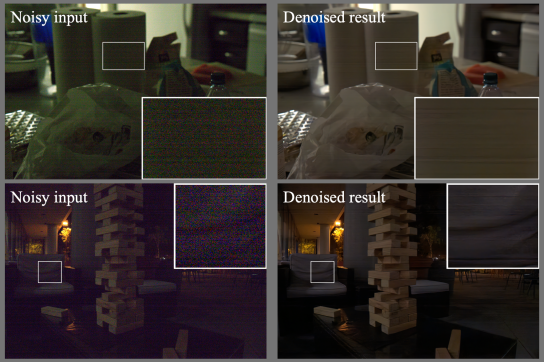


Figure: Denoising Results from a network trained on synthetic noise

Denoising Results

Testing our model: synthetize noise / add it to real images / Train a denoising Unet with this data

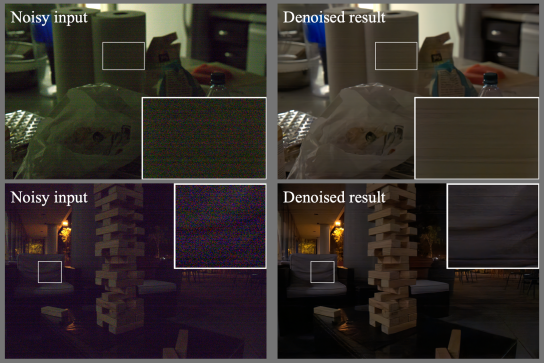
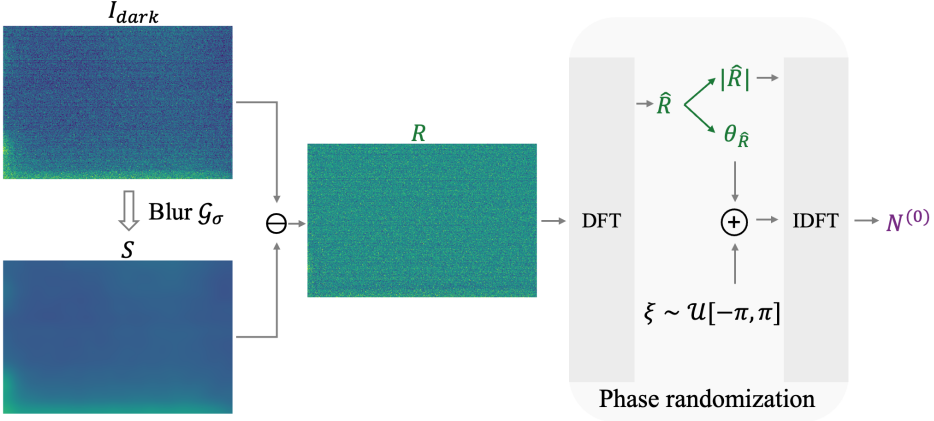


Figure: Denoising Results from a network trained on synthetic noise

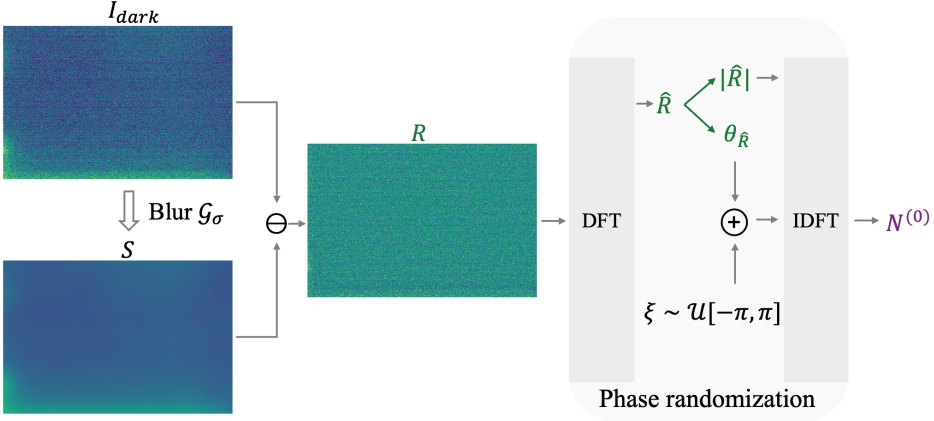
Problems: Banding pattern noise artifacts

Cause: No modeling of inter-channel correlations of noise



$$\xi \sim \mathcal{U}([-\pi, \pi]^{(C,H,W)})$$

Cause: No modeling of inter-channel correlations of noise



$\xi \sim \mathcal{U}([-\pi, \pi]^{(C,H,W)})$

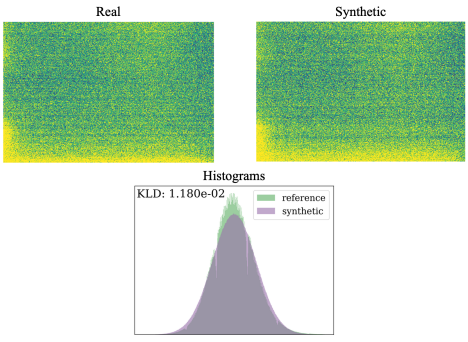
$\xi^0 \sim \mathcal{U}\left([-\pi, \pi]^{(H,W)}\right), \xi = replicate(\xi^0, C)$

Denoising Results

Figure: Denoising Results from a network trained on synthetic noise

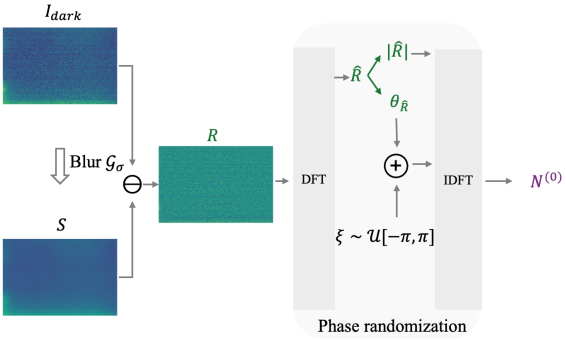
Histogram misalignment

RPN normalizes the noise histograms! $\rightarrow$

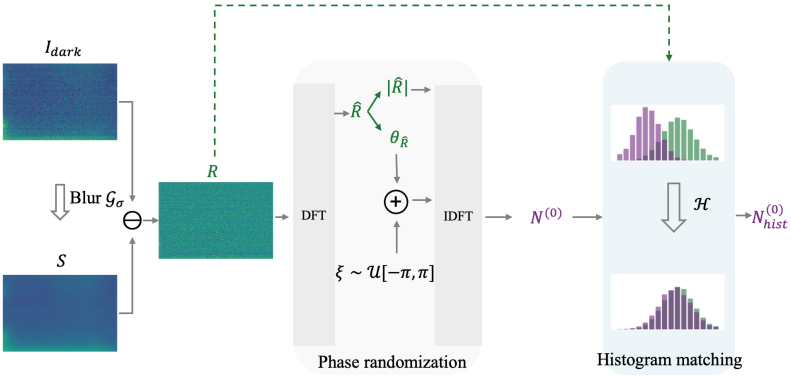


Color artifacts in denoising models

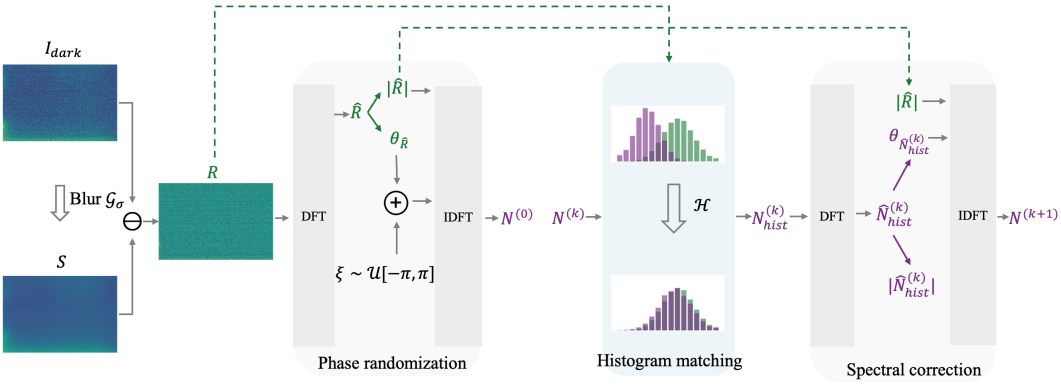
Histogram matching



Histogram matching



Histogram matching



Repeat (Histogram matching / Power spectrum prescription) K times

Denoising Results

No more color bleeding!

Figure: Denoising Results from a network trained on synthetic noise

Denoising Results

Better restoration of malfunctionning saturated pixels!

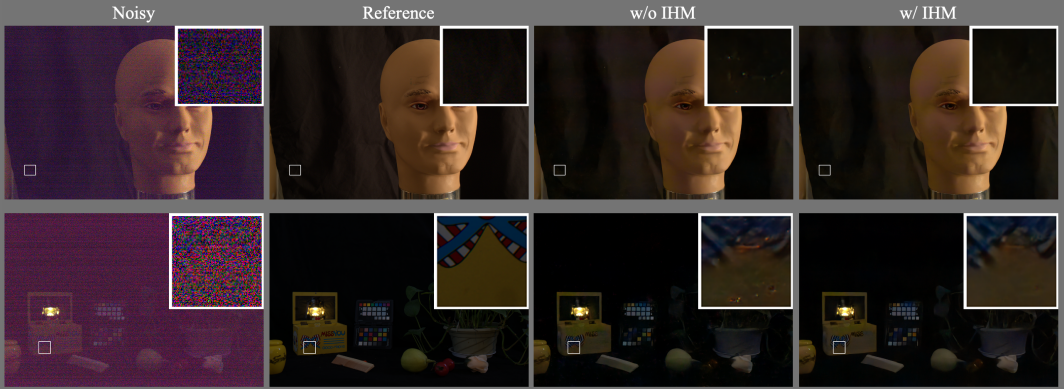
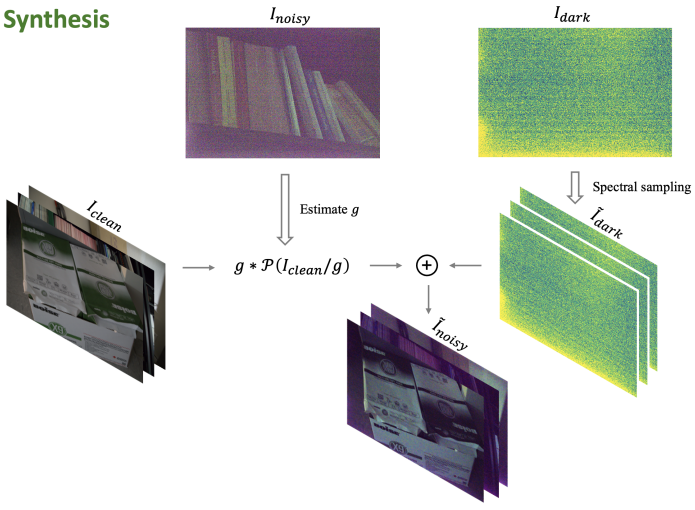


Figure: Denoising Results from a network trained on synthetic noise


Synthesis
I_{noisy}
I_{dark}
Estimate g
Spectral sampling
I_{clean}
$g * \mathcal{P}(I_{clean}/g)$
$\tilde{I}_{dark}$
$\tilde{I}_{noisy}$

EXPERIMENTAL RESULTS

Numerical Results

Dataset	Ratio	*Real data*	LRD [40]	NoiseDiff [29]	PG	ELD [38]	SFRN [45]	PMN [12]	Ours
SID	×100	*42.95 / 0.958*	43.16 / 0.958	**43.92 / 0.961**	41.05 / 0.936	41.95 / 0.953	42.81 / 0.957	43.47 / **0.961**	43.57 / **0.961**
	×250	*40.27 / 0.943*	40.69 / 0.941	**41.28** / 0.946	36.63 / 0.885	39.44 / 0.931	40.18 / 0.934	41.04 / **0.947**	41.24 / 0.945
	×300	*37.32 / 0.928*	37.48 / 0.919	**37.90** / 0.929	33.34 / 0.811	36.36 / 0.911	37.09 / 0.918	37.87 / **0.934**	37.77 / 0.929
ELD	×100	*45.52 / 0.977*	46.16 / 0.983	46.95 / 0.978	44.28 / 0.936	45.45 / 0.975	46.38 / 0.979	46.99 / 0.984	**47.13 / 0.986**
	×200	*41.70 / 0.912*	43.91 / 0.968	**45.11 / 0.971**	41.16 / 0.885	43.43 / 0.954	44.38 / 0.965	44.85 / 0.969	44.89 / 0.969

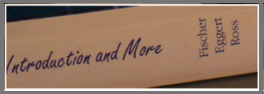
Figure: Denoising results on different benchmarks for low light denoising I

Ratio	*Real data*	PG	ELD [38]	SFRN [45]	PMN [12]	Ours
×64	*48.80 / 0.991*	46.98 / 0.988	48.26 / 0.990	48.18 / 0.990	49.32 / **0.992**	**49.47 / 0.992**
×128	*47.10 / 0.986*	45.93 / 0.982	46.69 / 0.984	46.75 / 0.986	47.60 / **0.987**	**47.72 / 0.987**
×256	*44.89 / 0.979*	44.09 / 0.970	44.47 / 0.974	44.84 / 0.979	45.41 / **0.981**	**45.50** / 0.979
×512	*42.59 / 0.966*	41.55 / 0.946	41.78 / 0.947	42.69 / 0.966	43.14 / **0.967**	**43.23** / 0.966
×1024	*40.29 / 0.945*	38.22 / 0.894	38.39 / 0.903	40.38 / 0.947	40.67 / **0.948**	**40.76** / 0.940
×64	*45.85 / 0.988*	42.51 / 0.980	45.09 / 0.984	45.18 / 0.985	**46.32 / 0.988**	46.31 / **0.988**
×128	*44.52 / 0.982*	41.78 / 0.972	43.63 / 0.974	43.83 / 0.977	**44.90 / 0.983**	44.76 / 0.980
×256	*42.71 / 0.971*	40.59 / 0.953	41.52 / 0.948	42.08 / 0.961	**43.01 / 0.970**	42.72 / 0.958

Figure: Denoising results on different benchmarks for low light denoising II

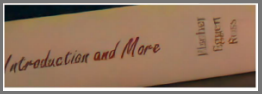
Visual Results

Noisy

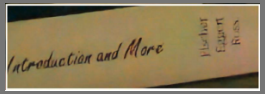


Clean

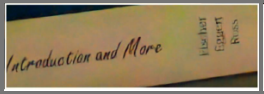
Real Pairs



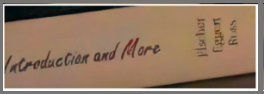
Poisson Gaussian



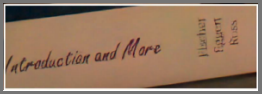
ELD



LRD

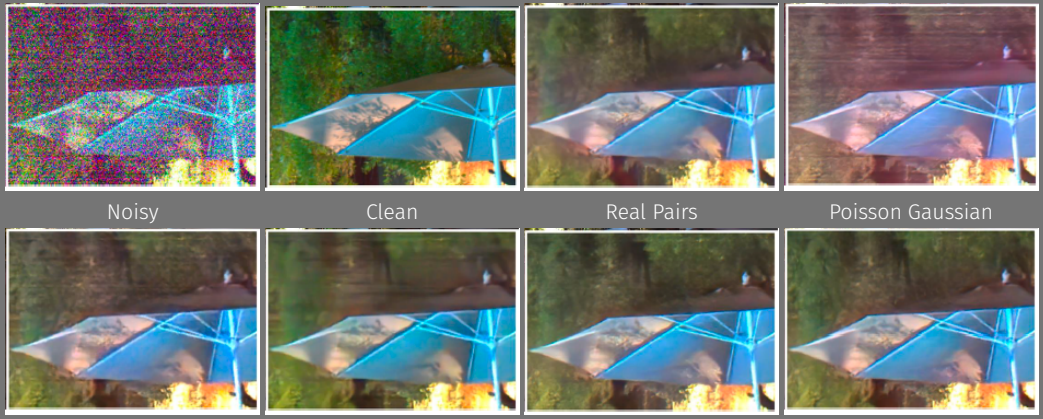


DarkDiff

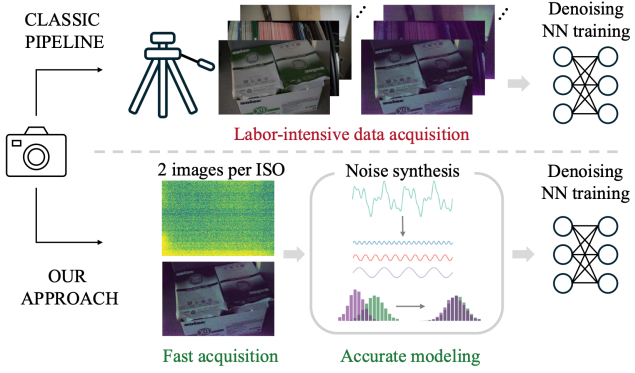


Ours

Visual Results



Overview



- minimal data requirements for model calibration
- no parametric estimation for signal dependent noise → less prone to errors
- SOTA image denoising results for low-light images across multiple sensors

Part III: Conclusion

2 sides of the same coin

Vibrant Leaves

$$\hat{\mathrm{p}}_{\mathsf{data}} \rightarrow \mathrm{p}_{\mathsf{VibrantLeaves}}$$

- A new model based on simple image properties
- Comparable image restoration performance with training on natural images
- A more interpretable model (prior visualization / model ablations)

Perspective:

- make the model differentiable $\rightarrow$ optimal parameters
- adapt for scientific computational imaging
- use $p_{\mathsf{VibrantLeaves}}$ as a p_0 for Flow Matching models

2 sides of the same coin

Vibrant Leaves

$$\hat{\mathrm{P}}_{\mathrm{data}} \rightarrow \mathrm{P}_{\mathrm{VibrantLeaves}}$$

- A new model based on simple image properties
- Comparable image restoration performance with training on natural images
- A more interpretable model (prior visualization / model ablations)

Perspective:

- make the model differentiable $\rightarrow$ optimal parameters
- adapt for scientific computational imaging
- use $p_{\mathrm{VibrantLeaves}}$ as a p_0 for Flow Matching models

part II

$$\mathrm{N}(0, \sigma^2) \rightarrow \text{2 Shots in the Dark}$$

A novel camera noise generator!

- a frugal, fast and easily portable methods for accurate noise synthesis
- no parametric estimation for signal independent noise
- state of the art denoising results

Perspective:

- account for additional noise factors (exposure time, sensor heat ...)
- deploy on other sensors in computational imaging (SPADs, CT scanners ...)

Collaborators and related publications

- Prof. Yann Gousseau (Télécom Paris, LTCI)
- Prof. Saïd Ladjal (Télécom Paris, LTCI)
- **Liying Lu**, PHD student (EPFL, IVRL)
- Prof. Sabine Süsstrunk (EPFL, IVRL)

Publications related to this talk:

- VibrantLeaves: A principled parametric image generator for training deep restoration models, *preprint 2025, RA, Y.Gousseau, S.Ladjal, S.Süsstrunk*
- 2-Shots in the Dark: Low-Light Denoising with Minimal Data Acquisition *preprint 2025, Liying Lu, RA, S.Süsstrunk*

Bibliography I


Abdelhamed, A., Brubaker, M. A., and Brown, M. S. (2019).
Noise flow: Noise modeling with conditional normalizing flows.
In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3165–3173.


Achddou, R., Gousseau, Y., and Ladjal, S. (2021).
Synthetic images as a regularity prior for image restoration neural networks.
In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 333–345. Springer.


Baradad, M., Wulff, J., Wang, T., Isola, P., and Torralba, A. (2021).
Learning to see by looking at noise.
Advances in Neural Information Processing Systems, 34.


Galerne, B., Gousseau, Y., and Morel, J.-M. (2010).
Random phase textures: Theory and synthesis.
IEEE Transactions on image processing, 20(1):257–267.


Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. (2017).
Clevr: A diagnostic dataset for compositional language and elementary visual reasoning.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910.


Kadkhodaie, Z. and Simoncelli, E. (2021).
Stochastic solutions for linear inverse problems using the prior implicit in a denoiser.
Advances in Neural Information Processing Systems, 34:13242–13254.


Kataoka, H., Okayasu, K., Matsumoto, A., Yamagata, E., Yamada, R., Inoue, N., Nakamura, A., and Satoh, Y. (2020).
Pre-training without natural images.
In *Proceedings of the Asian Conference on Computer Vision*.


Leclaire, A., Guez, E., and Galerne, B. (2025).
Backward Diffusion iterates Noising-Relaxed Denoising.
working paper or preprint.

Bibliography III


Lu, L., Achddou, R., and Susstrunk, S. (2025).
Dark noise diffusion: Noise synthesis for low-light image denoising.
IEEE Transactions on Pattern Analysis and Machine Intelligence.


Richter, S. R., Vineet, V., Roth, S., and Koltun, V. (2016).
Playing for data: Ground truth from computer games.
In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 102–118. Springer.


Wei, K., Fu, Y., Zheng, Y., and Yang, J. (2021).
Physics-based noise modeling for extreme low-light photography.
IEEE Transactions on Pattern Analysis and Machine Intelligence.


Zhang, F., Xu, B., Li, Z., Liu, X., Lu, Q., Gao, C., and Sang, N. (2023).
Towards general low-light raw noise synthesis and modeling.
In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10820–10830.

Bibliography IV


Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L., and Timofte, R. (2021).
Plug-and-play image restoration with deep denoiser prior.
IEEE Transactions on Pattern Analysis and Machine Intelligence.


Zhang, K., Zuo, W., and Zhang, L. (2018).
Ffdnet: Toward a fast and flexible solution for cnn-based image denoising.
IEEE Transactions on Image Processing, 27(9):4608–4622.