# Class-specific Poisson Image Denoising using Importance Sampling

Milad Niknejad, José M. Bioucas-Dias, Mário A. T. Figueiredo

Instituto de Telecomunicações, Instituto Superior Técnico,
University of Lisbon, Portugal

SpaRTaN
Sparse Representations and Compressed Sensing Training Network

# Overview

1. Class-specific image denoising
2. Patch estimation using Monte-Carlo
3. Importance Sampling
4. Applying importance sampling for image denoising
5. Proposed method

# Class-specific image denoising

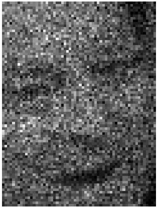
Often, the image to be denoised belongs to a known specific class,

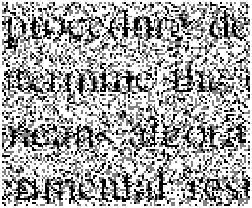
*Examples:* text/document, face, fingerprint, a specific type of medical image (*e.g.*, brain MRI), ...

# Class-specific image denoising

Often, the image to be denoised belongs to a known specific class,

*Examples:* text/document, face, fingerprint, a specific type of medical image (*e.g.*, brain MRI), ...

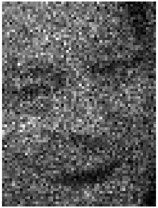


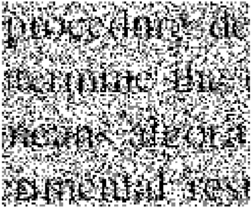


# Class-specific image denoising

Often, the image to be denoised belongs to a known specific class,

*Examples:* text/document, face, fingerprint, a specific type of medical image (*e.g.*, brain MRI), ...





This knowledge should be exploited by the denoising method!

# Class-specific image denoising

Assumption: A dataset of clean images of the same class is available.

# Patch-based image denoising

Gaussian noise observation model:

$$\mathbf{y}_i = \mathbf{x}_i + \mathbf{v}_i$$

$\mathbf{x}_i$ is a patch of the original image; $\mathbf{y}_i$ is the corresponding noisy patch; $\mathbf{v}_i$ is i.i.d. Gaussian noise.

# Patch-based image denoising

Gaussian noise observation model:

$$\mathbf{y}_i = \mathbf{x}_i + \mathbf{v}_i$$

$\mathbf{x}_i$ is a patch of the original image; $\mathbf{y}_i$ is the corresponding noisy patch;
$\mathbf{v}_i$ is i.i.d. Gaussian noise.

Poisson noise observation model (the focus of this presentation):

$$\mathbf{y}_{i,j} \sim \mathcal{P}(\mathbf{x}_{i,j}).$$

$\mathbf{x}_{i,j}$ is the $j^{th}$ pixel of $\mathbf{x}_i$.
$\mathcal{P}$ is a Poisson distribution with mean $\mathbf{x}_{i,j}$.

# Patch-based image denoising

Gaussian noise observation model:

$$\mathbf{y}_i = \mathbf{x}_i + \mathbf{v}_i$$

$\mathbf{x}_i$ is a patch of the original image; $\mathbf{y}_i$ is the corresponding noisy patch; $\mathbf{v}_i$ is i.i.d. Gaussian noise.

Poisson noise observation model (the focus of this presentation):

$$\mathbf{y}_{i,j} \sim \mathcal{P}(\mathbf{x}_{i,j}).$$

$\mathbf{x}_{i,j}$ is the $j^{th}$ pixel of $\mathbf{x}_i$.
$\mathcal{P}$ is a Poisson distribution with mean $\mathbf{x}_{i,j}$.

**Goal**: recover the clean patch $\mathbf{x}_i$ from the noisy one $\mathbf{y}_i$

# MMSE (*minimum mean squared error*) estimation

MMSE patch estimate ($p(\mathbf{y} = \mathbf{y}_i)$ is replaced by $p(\mathbf{y}_i)$):

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\, p(\mathbf{x}|\mathbf{y}_i)\, d\mathbf{x} = \int_{\mathbb{R}^p} \mathbf{x}\, \frac{p(\mathbf{y}_i|\mathbf{x})\, p(\mathbf{x})}{p(\mathbf{y}_i)}\, d\mathbf{x}$$

# MMSE (*minimum mean squared error*) estimation

MMSE patch estimate ($p(\mathbf{y} = \mathbf{y}_i)$ is replaced by $p(\mathbf{y}_i)$):

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\, p(\mathbf{x}|\mathbf{y}_i)\, d\mathbf{x} = \int_{\mathbb{R}^p} \mathbf{x}\, \frac{p(\mathbf{y}_i|\mathbf{x})\, p(\mathbf{x})}{p(\mathbf{y}_i)}\, d\mathbf{x}$$

- This multi-dimensional integral is intractable, in general (exception: Gaussian noise and Gaussian prior).

# MMSE (*minimum mean squared error*) estimation

MMSE patch estimate ($p(\mathbf{y} = \mathbf{y}_i)$ is replaced by $p(\mathbf{y}_i)$):

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\, p(\mathbf{x}|\mathbf{y}_i)\, d\mathbf{x} = \int_{\mathbb{R}^p} \mathbf{x}\, \frac{p(\mathbf{y}_i|\mathbf{x})\, p(\mathbf{x})}{p(\mathbf{y}_i)}\, d\mathbf{x}$$

- This multi-dimensional integral is intractable, in general (exception: Gaussian noise and Gaussian prior).
- Monte-Carlo approximation: obtain samples $\mathbf{x}_j$ from $p(\mathbf{x}|\mathbf{y})$

# MMSE (*minimum mean squared error*) estimation

MMSE patch estimate ($p(\mathbf{y} = \mathbf{y}_i)$ is replaced by $p(\mathbf{y}_i)$):

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\; p(\mathbf{x}|\mathbf{y}_i)\; d\mathbf{x} = \int_{\mathbb{R}^p} \mathbf{x}\; \frac{p(\mathbf{y}_i|\mathbf{x})\, p(\mathbf{x})}{p(\mathbf{y}_i)}\; d\mathbf{x}$$

- This multi-dimensional integral is intractable, in general (exception: Gaussian noise and Gaussian prior).
- Monte-Carlo approximation: obtain samples $\mathbf{x}_j$ from $p(\mathbf{x}|\mathbf{y})$

$$\hat{\hat{\mathbf{x}}}_i = \frac{1}{n}\sum_{j=1}^{n} \mathbf{x}_j \qquad \lim_{n\to\infty} \hat{\hat{\mathbf{x}}}_i = \hat{\mathbf{x}}_i$$

# MMSE (*minimum mean squared error*) estimation

MMSE patch estimate ($p(\mathbf{y} = \mathbf{y}_i)$ is replaced by $p(\mathbf{y}_i)$):

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\, p(\mathbf{x}|\mathbf{y}_i)\, d\mathbf{x} = \int_{\mathbb{R}^p} \mathbf{x}\, \frac{p(\mathbf{y}_i|\mathbf{x})\, p(\mathbf{x})}{p(\mathbf{y}_i)}\, d\mathbf{x}$$

- This multi-dimensional integral is intractable, in general (exception: Gaussian noise and Gaussian prior).
- Monte-Carlo approximation: obtain samples $\mathbf{x}_j$ from $p(\mathbf{x}|\mathbf{y})$

$$\hat{\hat{\mathbf{x}}}_i = \frac{1}{n}\sum_{j=1}^{n} \mathbf{x}_j \qquad \lim_{n\to\infty} \hat{\hat{\mathbf{x}}}_i = \hat{\mathbf{x}}_i$$

- However, sampling from $p(\mathbf{x}|\mathbf{y}_i)$ is also intractable.

# MMSE (*minimum mean squared error*) estimation

MMSE patch estimate ($p(\mathbf{y} = \mathbf{y}_i)$ is replaced by $p(\mathbf{y}_i)$):

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\, p(\mathbf{x}|\mathbf{y}_i)\, d\mathbf{x} = \int_{\mathbb{R}^p} \mathbf{x}\, \frac{p(\mathbf{y}_i|\mathbf{x})\, p(\mathbf{x})}{p(\mathbf{y}_i)}\, d\mathbf{x}$$

- This multi-dimensional integral is intractable, in general (exception: Gaussian noise and Gaussian prior).
- Monte-Carlo approximation: obtain samples $\mathbf{x}_j$ from $p(\mathbf{x}|\mathbf{y})$

$$\hat{\hat{\mathbf{x}}}_i = \frac{1}{n}\sum_{j=1}^{n} \mathbf{x}_j \qquad \lim_{n\to\infty} \hat{\hat{\mathbf{x}}}_i = \hat{\mathbf{x}}_i$$

- However, sampling from $p(\mathbf{x}|\mathbf{y}_i)$ is also intractable.
- Can we approximate $\hat{\mathbf{x}}_i$ by sampling from another distribution?

# Self-Normalized Importance Sampling (SNIS)

Goal: to compute (or approximate)

$$\mathbb{E}[f(\mathbf{z})] = \int f(\mathbf{z})\, p(\mathbf{z})\; d\mathbf{z}.$$

# Self-Normalized Importance Sampling (SNIS)

Goal: to compute (or approximate)

$$\mathbb{E}[f(\mathbf{z})] = \int f(\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}.$$

- Let $\tilde{p}(\mathbf{z}) = c\, p(\mathbf{z})$ be an un-normalized version of $p(\mathbf{z})$.

# Self-Normalized Importance Sampling (SNIS)

Goal: to compute (or approximate)

$$\mathbb{E}[f(\mathbf{z})] = \int f(\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}.$$

- Let $\tilde{p}(\mathbf{z}) = c\, p(\mathbf{z})$ be an un-normalized version of $p(\mathbf{z})$.
- Let $\tilde{q}(\mathbf{z}) = b\, q(\mathbf{z})$ be another un-normalized density; assume it is possible/easy to obtain samples $\mathbf{z}_1, ..., \mathbf{z}_n \sim q(\mathbf{z})$.
- Constants $c$ and $b$ may be unknown.

# Self-Normalized Importance Sampling (SNIS)

Goal: to compute (or approximate)

$$\mathbb{E}[f(\mathbf{z})] = \int f(\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}.$$

- Let $\tilde{p}(\mathbf{z}) = c\, p(\mathbf{z})$ be an un-normalized version of $p(\mathbf{z})$.
- Let $\tilde{q}(\mathbf{z}) = b\, q(\mathbf{z})$ be another un-normalized density; assume it is possible/easy to obtain samples $\mathbf{z}_1, ..., \mathbf{z}_n \sim q(\mathbf{z})$.
- Constants $c$ and $b$ may be unknown.

$$\hat{\mathbb{E}}_n[f(\mathbf{z})] = \frac{\sum_{j=1}^{n} f(\mathbf{z}_j) w(\mathbf{z}_j)}{\sum_{j=1}^{n} w(\mathbf{z}_j)}, \qquad w(\mathbf{z}_j) = \frac{\tilde{p}(\mathbf{z}_j)}{\tilde{q}(\mathbf{z}_j)}.$$

# Self-Normalized Importance Sampling (SNIS)

Goal: to compute (or approximate)

$$\mathbb{E}[f(\mathbf{z})] = \int f(\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}.$$

- Let $\tilde{p}(\mathbf{z}) = c\, p(\mathbf{z})$ be an un-normalized version of $p(\mathbf{z})$.
- Let $\tilde{q}(\mathbf{z}) = b\, q(\mathbf{z})$ be another un-normalized density; assume it is possible/easy to obtain samples $\mathbf{z}_1, ..., \mathbf{z}_n \sim q(\mathbf{z})$.
- Constants $c$ and $b$ may be unknown.

$$\hat{\mathbb{E}}_n[f(\mathbf{z})] = \frac{\sum_{j=1}^{n} f(\mathbf{z}_j) w(\mathbf{z}_j)}{\sum_{j=1}^{n} w(\mathbf{z}_j)}, \qquad w(\mathbf{z}_j) = \frac{\tilde{p}(\mathbf{z}_j)}{\tilde{q}(\mathbf{z}_j)}.$$

- As in plain Monte-Carlo: $\lim_{n\to\infty} \hat{\mathbb{E}}_n[f(\mathbf{z})] = \mathbb{E}[f(\mathbf{z})]$

# Applying SNIS for MMSE patch estimation

Back to our problem:

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\ p(\mathbf{x}|\mathbf{y}_i)\ d\mathbf{x}.$$

# Applying SNIS for MMSE patch estimation

Back to our problem:

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\; p(\mathbf{x}|\mathbf{y}_i)\; d\mathbf{x}.$$

- Instead of sampling from $p(\mathbf{x}|\mathbf{y}_i)$, use samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from $p(\mathbf{x})$;

# Applying SNIS for MMSE patch estimation

Back to our problem:

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x} \, p(\mathbf{x}|\mathbf{y}_i) \, d\mathbf{x}.$$

- Instead of sampling from $p(\mathbf{x}|\mathbf{y}_i)$, use samples $\mathbf{x}_1, ..., \mathbf{x}_n$ from $p(\mathbf{x})$;
- Simply use samples from the external dataset of clean patches.

# Applying SNIS for MMSE patch estimation

Back to our problem:

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\; p(\mathbf{x}|\mathbf{y}_i)\; d\mathbf{x}.$$

- Instead of sampling from $p(\mathbf{x}|\mathbf{y}_i)$, use samples $\mathbf{x}_1, ..., \mathbf{x}_n$ from $p(\mathbf{x})$;
- Simply use samples from the external dataset of clean patches.
- Use these samples in SNIS

$$\hat{\hat{\mathbf{x}}}_i = \hat{\mathbb{E}}_n[\mathbf{x}|\mathbf{y}_i] = \frac{\sum_{j=1}^{n} \mathbf{x}_j\, w_j}{\sum_{j=1}^{n} w_j}, \qquad w_j = p(\mathbf{y}_i|\mathbf{x} = \mathbf{x}_j)$$

# Applying SNIS for MMSE patch estimation

Back to our problem:

$$\hat{\mathbf{x}}_i = \mathbb{E}[\mathbf{x}|\mathbf{y}_i] = \int_{\mathbb{R}^p} \mathbf{x}\; p(\mathbf{x}|\mathbf{y}_i)\; d\mathbf{x}.$$

- Instead of sampling from $p(\mathbf{x}|\mathbf{y}_i)$, use samples $\mathbf{x}_1, ..., \mathbf{x}_n$ from $p(\mathbf{x})$;
- Simply use samples from the external dataset of clean patches.
- Use these samples in SNIS

$$\hat{\hat{\mathbf{x}}}_i = \hat{\mathbb{E}}_n[\mathbf{x}|\mathbf{y}_i] = \frac{\displaystyle\sum_{j=1}^{n} \mathbf{x}_j\, w_j}{\displaystyle\sum_{j=1}^{n} w_j}, \qquad w_j = p(\mathbf{y}_i|\mathbf{x} = \mathbf{x}_j)$$

- Why? $\tilde{p}(\mathbf{z}) = p(\mathbf{y}_i|\mathbf{x})\, p(\mathbf{x})$, $c = 1/p(\mathbf{y}_i)$ and $\tilde{q}(\mathbf{z}) = p(\mathbf{x})$.

# Applying SNIS for MMSE patch estimation (II)

- For Poisson noise, the weights are easy to obtain ($\mathbf{y}_{i,j} \sim \mathcal{P}(\mathbf{x}_{i,j})$, i.i.d.)

$$w_j = \prod_{l=1}^{N} \frac{e^{-\mathbf{x}_{(j,l)}} \left(\mathbf{x}_{(j,l)}\right)^{\mathbf{y}_{(j,l)}}}{\mathbf{y}_{(j,l)}!}$$

# Applying SNIS for MMSE patch estimation (II)

- For Poisson noise, the weights are easy to obtain ($\mathbf{y}_{i,j} \sim \mathcal{P}(\mathbf{x}_{i,j})$, i.i.d.)

$$w_j = \prod_{l=1}^{N} \frac{e^{-\mathbf{x}_{(j,l)}} (\mathbf{x}_{(j,l)})^{\mathbf{y}_{(j,l)}}}{\mathbf{y}_{(j,l)}!}$$

- It can be adapted to other image restoration tasks, such as deblurring,

$$w_j = \prod_{l=1}^{N} \frac{e^{-\mathbf{H}_{(j,l)}\mathbf{x}_{(j,l)}} (\mathbf{H}_{(j,l)}\mathbf{x}_{(j,l)})^{\mathbf{y}_{(j,l)}}}{\mathbf{y}_{(j,l)}!}$$

# Applying SNIS for MMSE patch estimation (II)

- For Poisson noise, the weights are easy to obtain ($\mathbf{y}_{i,j} \sim \mathcal{P}(\mathbf{x}_{i,j})$, i.i.d.)

$$w_j = \prod_{l=1}^{N} \frac{e^{-\mathbf{x}_{(j,l)}} (\mathbf{x}_{(j,l)})^{\mathbf{y}_{(j,l)}}}{\mathbf{y}_{(j,l)}!}$$

- It can be adapted to other image restoration tasks, such as deblurring,

$$w_j = \prod_{l=1}^{N} \frac{e^{-\mathbf{H}_{(j,l)}\mathbf{x}_{(j,l)}} (\mathbf{H}_{(j,l)}\mathbf{x}_{(j,l)})^{\mathbf{y}_{(j,l)}}}{\mathbf{y}_{(j,l)}!}$$

- It can be generalized to other noise models.

# Making it work: proposed approach

Key observations:

1. Using samples from $p(\mathbf{x})$ is sub-optimal, as it may have high variance (or even infinite variance). It requires very large $n$.

# Making it work: proposed approach

Key observations:

1. Using samples from $p(\mathbf{x})$ is sub-optimal, as it may have high variance (or even infinite variance). It requires very large $n$.

2. The proposal distribution should be made similar to each target distribution $p(\mathbf{x}|\mathbf{y}_i)$: Estimator with lower MMSE for limited number of samples.

# Making it work: proposed approach

Key observations:

1. Using samples from $p(\mathbf{x})$ is sub-optimal, as it may have high variance (or even infinite variance). It requires very large $n$.

2. The proposal distribution should be made similar to each target distribution $p(\mathbf{x}|\mathbf{y}_i)$: Estimator with lower MMSE for limited number of samples.

Proposed approach:

1. Cluster the patches in the external dataset.

# Making it work: proposed approach

Key observations:

1. Using samples from $p(\mathbf{x})$ is sub-optimal, as it may have high variance (or even infinite variance). It requires very large $n$.
2. The proposal distribution should be made similar to each target distribution $p(\mathbf{x}|\mathbf{y}_i)$: Estimator with lower MMSE for limited number of samples.

Proposed approach:

1. Cluster the patches in the external dataset.
2. Assign each noisy patch to the closest cluster.

# Making it work: proposed approach

Key observations:

1. Using samples from $p(\mathbf{x})$ is sub-optimal, as it may have high variance (or even infinite variance). It requires very large $n$.

2. The proposal distribution should be made similar to each target distribution $p(\mathbf{x}|\mathbf{y}_i)$: Estimator with lower MMSE for limited number of samples.

Proposed approach:

1. Cluster the patches in the external dataset.

2. Assign each noisy patch to the closest cluster.

3. Use the corresponding clean patches as samples from the proposal distribution for SNIS.

# Proposed method

- Clustering: Any clustering algorithm can be used (k-means,...). The whole dataset of patches is clustered to $K$ clusters. $\{\mathbf{X}_1 \ldots \mathbf{X}_K\}$.

# Proposed method

- Clustering: Any clustering algorithm can be used (k-means,...). The whole dataset of patches is clustered to $K$ clusters. $\{\mathbf{X}_1 \ldots \mathbf{X}_K\}$.

- Objective is to solve the following simultaneous classification and estimation problem:

$$(\hat{\mathbf{x}}_i, \hat{k}_i) = \arg\min_{(\mathbf{u},k)} \int_{\mathbb{R}_+^m} \|\mathbf{u} - \mathbf{x}\|_2^2 \, p(\mathbf{x}|\mathbf{y}_i, k) \; d\mathbf{x}$$

# Proposed method

- Clustering: Any clustering algorithm can be used (k-means,...). The whole dataset of patches is clustered to $K$ clusters. $\{\mathbf{X}_1 \dots \mathbf{X}_K\}$.
- Objective is to solve the following simultaneous classification and estimation problem:

$$(\hat{\mathbf{x}}_i, \hat{k}_i) = \arg\min_{(\mathbf{u},k)} \int_{\mathbb{R}_+^m} \|\mathbf{u} - \mathbf{x}\|_2^2 \; p(\mathbf{x}|\mathbf{y}_i, k) \; d\mathbf{x}$$

- The above chooses the best cluster $\hat{k}_i$, and use this distribution to approximate the integral.

# Proposed method

- Clustering: Any clustering algorithm can be used (k-means,...).
  The whole dataset of patches is clustered to $K$ clusters. $\{\mathbf{X}_1 \dots \mathbf{X}_K\}$.

- Objective is to solve the following simultaneous classification and estimation problem:

$$(\hat{\mathbf{x}}_i, \hat{k}_i) = \arg\min_{(\mathbf{u},k)} \int_{\mathbb{R}^m_+} \|\mathbf{u} - \mathbf{x}\|_2^2 \; p(\mathbf{x}|\mathbf{y}_i, k) \;\; d\mathbf{x}$$

- The above chooses the best cluster $\hat{k}_i$, and use this distribution to approximate the integral.

- It is equivalent to sampling from (unkown) $\hat{k}_i^{th}$ distribution as the proposal distribution.

# Proposed method

- Clustering: Any clustering algorithm can be used (k-means,...). The whole dataset of patches is clustered to $K$ clusters. $\{\mathbf{X}_1 \dots \mathbf{X}_K\}$.

- Objective is to solve the following simultaneous classification and estimation problem:

$$(\hat{\mathbf{x}}_i, \hat{k}_i) = \underset{(\mathbf{u},k)}{\arg\min} \int_{\mathbb{R}^m_+} \|\mathbf{u} - \mathbf{x}\|_2^2 \; p(\mathbf{x}|\mathbf{y}_i, k) \;\; d\mathbf{x}$$

- The above chooses the best cluster $\hat{k}_i$, and use this distribution to approximate the integral.

- It is equivalent to sampling from (unkown) $\hat{k}_i^{th}$ distribution as the proposal distribution.

- The above integral is intractable, but we can use SNIS.

# Proposed method

$$\mathbb{E}[\|\mathbf{x}-\mathbf{u}\|_2^2|\mathbf{y}_i,k] = \int_{\mathbb{R}_+^m} \|\mathbf{u}-\mathbf{x}\|_2^2 \; p(\mathbf{x}|\mathbf{y}_i,k) \; d\mathbf{x}.$$

# Proposed method

$$\mathbb{E}[\|\mathbf{x}-\mathbf{u}\|_2^2|\mathbf{y}_i,k] = \int_{\mathbb{R}_+^m} \|\mathbf{u}-\mathbf{x}\|_2^2 \; p(\mathbf{x}|\mathbf{y}_i,k)\; d\mathbf{x}.$$

Using SNIS, the above can be approximated by

$$\hat{\mathbb{E}}_n[\|\mathbf{x}-\mathbf{u}\|_2^2|\mathbf{y}_i,k] = \frac{\sum_{j=1}^{n} \|\mathbf{u}-\mathbf{x}_j\|_2^2 \; p(\mathbf{y}_i|\mathbf{x}_{j_k})}{\sum_{j=1}^{n} p(\mathbf{y}_i|\mathbf{x}_{j_k})} \tag{1}$$

where the $\mathbf{x}_{j_k}$, for $j=1,...,n$ are samples from the distribution $p(\mathbf{x}|k)$.

# Proposed method

We minimize the approximation (1) by alternating minimization.

$$(\hat{\mathbf{x}}_i, \hat{k}_i) = \arg\min_{(\mathbf{u},k)} \hat{\mathbb{E}}_n[\|\mathbf{x} - \mathbf{u}\|_2^2 | \mathbf{y}_i, k]$$

# Proposed method

We minimize the approximation (1) by alternating minimization.

$$(\hat{\mathbf{x}}_i, \hat{k}_i) = \arg\min_{(\mathbf{u},k)} \hat{\mathbb{E}}_n[\|\mathbf{x} - \mathbf{u}\|_2^2 | \mathbf{y}_i, k]$$

- when $\mathbf{u}$ is fixed,

$$\hat{k}_i(\mathbf{u}) = \arg\min_k \frac{\sum_{j=1}^{n_2} \|\mathbf{u} - \mathbf{x}_j\|_2^2 \, p(\mathbf{y}_i | \mathbf{x}_{j_k})}{\sum_{j=1}^{n_2} p(\mathbf{y}_i | \mathbf{x}_{j_k})}. \tag{2}$$

# Proposed method

We minimize the approximation (1) by alternating minimization.

$$(\hat{\mathbf{x}}_i, \hat{k}_i) = \underset{(\mathbf{u},k)}{\arg\min}\, \hat{\mathbb{E}}_n[\|\mathbf{x}-\mathbf{u}\|_2^2 | \mathbf{y}_i, k]$$

- when $\mathbf{u}$ is fixed,

$$\hat{k}_i(\mathbf{u}) = \arg\min_k \frac{\sum_{j=1}^{n_2} \|\mathbf{u}-\mathbf{x}_j\|_2^2\, p(\mathbf{y}_i|\mathbf{x}_{j_k})}{\sum_{j=1}^{n_2} p(\mathbf{y}_i|\mathbf{x}_{j_k})}. \tag{2}$$

- when $k$ is fixed,

$$\hat{\mathbf{x}}_i(k) = \hat{\mathbb{E}}_{n_1}[\mathbf{x}|\mathbf{y}_i, k] = \frac{\sum_{j=1}^{n_1} \mathbf{x}_j p(\mathbf{y}_i|\mathbf{x}_j)}{\sum_{j=1}^{n_1} P(\mathbf{y}_i|\mathbf{x}_j)} \tag{3}$$

# Implementation Details

Speeding up the algorithm:

# Implementation Details

Speeding up the algorithm:

- The key to speeding up is to limit the numbers of patch samples $n_1$ and $n_2$.

# Implementation Details

Speeding up the algorithm:

- The key to speeding up is to limit the numbers of patch samples $n_1$ and $n_2$.
- Clustering: $n_2 = 30$, overall 600 patches for all $k = 20$ clusters (less than 1 percent of samples in external datasets).

# Implementation Details

Speeding up the algorithm:

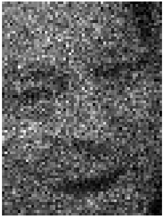
- The key to speeding up is to limit the numbers of patch samples $n_1$ and $n_2$.
- Clustering: $n_2 = 30$, overall 600 patches for all $k = 20$ clusters (less than 1 percent of samples in external datasets).
- Denoising: samples derived for each patch $n_1$ was set to 300.

# Implementation Details

Speeding up the algorithm:

- The key to speeding up is to limit the numbers of patch samples $n_1$ and $n_2$.
- Clustering: $n_2 = 30$, overall 600 patches for all $k = 20$ clusters (less than 1 percent of samples in external datasets).
- Denoising: samples derived for each patch $n_1$ was set to 300.
- Overall: 900 patches are processed for each denoised patch (computational complexity is similar to an internal non-local denoising with the patches constrained in $30 \times 30$ window).

# Experiment 1



Noisy image
(Peak=10)

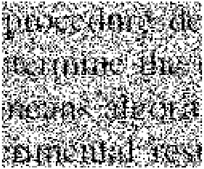
Non-local PCA
(PSNR=22.60)

VST+BM3D
(PSNR=24.79)

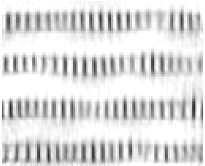
Poisson NL means
(PSNR=24.55)

Proposed
(PSNR=26.40)

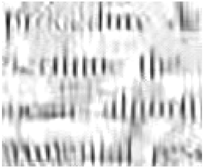
# Experiment 2



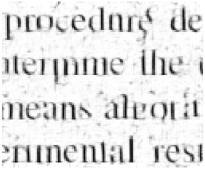
Noisy (Peak=2)



Non-local PCA (PSNR=14.95)



VST+BM3D (PSNR=14.55)



Proposed (PSNR=18.64)

# Conclusion

- We proposed a method based on importance sampling in which no parametric distribution is fitted to data

# Conclusion

- We proposed a method based on importance sampling in which no parametric distribution is fitted to data
- Any clustering method can be used

# Conclusion

- We proposed a method based on importance sampling in which no parametric distribution is fitted to data
- Any clustering method can be used
- Each cluster can be seen as samples of unknown proposal distribution

# Conclusion

- We proposed a method based on importance sampling in which no parametric distribution is fitted to data
- Any clustering method can be used
- Each cluster can be seen as samples of unknown proposal distribution
- The method can be generalized easily to other image restoration inverse problems.

# References


A. Owen
Monte Carlo Theory, Methods and Examples
Available at http://statweb.stanford.edu/ owen/mc/.

Bugallo, M.F., Martino, L. and Corander, J.
Adaptive importance sampling in signal processing
Digital Signal Processing 47 (2015): 36-49

Salmon, J., Harmany, Z., Deledalle, C. A., Willett, R.
Poisson noise reduction with non-local PCA.
Journal of mathematical imaging and vision, 48(2), 279-294.

Thanks for your attention