Video denoising via Bayesian modeling of patches

Pablo Arias
joint work with Thibaud Ehret and Jean-Michel Morel
CMLA, ENS Paris-Saclay

IHP workshop: Statistical modeling for shapes and imaging
Paris - 14/03/2019
Why are we still researching in denoising

Denoising is mainly for people (i.e. not as much as a pre-processing for a CV task)

- Photography
- Smartphone cameras
- Film industry
- Night vision cameras
- Surveillance cameras
- Medical imaging
- Astronomy

A lot of research has been devoted to still image denoising, but there is still room for improvement in video denoising.
Recursive vs. non-recursive methods

In this talk: video denoising via Bayesian estimation of patches, non-recursive and recursive approaches.
Recursive vs. non-recursive methods

In this talk: video denoising via Bayesian estimation of patches, non-recursive and recursive approaches.
State of the art

Non-recursive

- Non-local means [Buades et al.'05], [Liu, Freeman '10]
- K-SVD [Protter, Elad '07]
- V-BM3D [Dabov et al.'07]
- Graph regularization [Ghoniem et al. '10]
- BM4D, V-BM4D [Maggioni et al.'12]
- SPTWO [Buades et al.'17]
- VNLB [Arias, Morel'18]
- VIDOSAT [Wen et al.'19]
- SALT [Wen et al.'19]
- VNNet [Davy et al.'19]

Recursive

- Bilateral + Kalman filter [Zuo et al.'13]
- Bilateral + Kalman filter [Pfleger et al.'17]
- Recursive NL-means [Ali, Hardie'17]
- Gaussian-Laplacian fusion [Ehmann et al.'18]
State of the art

Non-recursive

- Non-local means [Buades et al.'05], [Liu, Freeman '10]
- K-SVD [Protter, Elad '07]
- V-BM3D [Dabov et al.'07]
- Graph regularization [Ghoniem et al. '10]
- BM4D
- SPTWO
- VNL
- VID
- SALT [Wen et al.'19]
- VNLnet [Davy et al.'19]

Recursive

- Bilateral + Kalman filter [Zuo et al.'13]
- Bilateral + Kalman filter [Pfleger et al.'17]
- Recursive NL-means [Ali, Hardie'17]
- Gaussian-Laplacian fusion [Ehmann et al.'18]

Non-recursive/mixed approaches good results at high computational cost.

Recursive methods: real-time performance but worse results.
Contents

Non-recursive methods

Recursive method I

Recursive method II

Empirical comparison
Contents

Non-recursive methods

Recursive method I

Recursive method II

Empirical comparison
Attention! Strange diagrams ahead
Video denoising framework

1. Extract $n$ similar patches
2. Estimate model parameters
3. Wiener filtering
4. Aggregate by averaging
Video denoising framework

Extract $n$ similar patches

$[q_1, \ldots, q_n] \rightarrow$ Wiener filtering $[\hat{p}_1, \ldots, \hat{p}_n] \rightarrow$ Aggregate by averaging

Estimate model parameters

$[q_1, \ldots, q_n]$
Video denoising framework

- Extract $n$ similar patches
- Wiener filtering
- Aggregate by averaging
- Estimate model parameters

$[q_1, \ldots, q_n] \rightarrow \hat{p}_1, \ldots, \hat{p}_n$
Video denoising framework

Extract $n$ similar patches → Wiener filtering → Aggregate by averaging

Estimate model parameters

$[q_1, \ldots, q_n]$ → $[\hat{p}_1, \ldots, \hat{p}_n]$
VNLB: Bayesian patch estimation with Gaussian prior

\( p_1, \ldots, p_n : n \) “similar” clean patches from \( u \)

\( q_1, \ldots, q_n : n \) “similar” noisy patches from \( v \)

**Assumption:** patches are IID samples of a Gaussian distribution

\[
P(p_i) = \mathcal{N}(p_i | \mu, C) \\
P(q_i | p_i) = \mathcal{N}(q_i | p_i, \sigma^2 I)
\]

For each noisy patch \( q_i \) estimate \( p_i \) as the MAP/MMSE, given by the Wiener filter:

\[
\hat{p}_i = \mu + C(C + \sigma^2 I)^{-1}(q_i - \mu)
\]
VNLB: Bayesian patch estimation with Gaussian prior

\( p_1, \ldots, p_n \): \( n \) “similar” clean patches from \( u \)

\( q_1, \ldots, q_n \): \( n \) “similar” noisy patches from \( v \)

Assumption: patches are IID samples of a Gaussian distribution

\[
P(p_i) = \mathcal{N}(p_i \mid \mu, C) \\
\mathbb{P}(q_i \mid p_i) = \mathcal{N}(q_i \mid p_i, \sigma^2 I)
\]

For each noisy patch \( q_i \) estimate \( p_i \) as the MAP/MMSE, given by the Wiener filter:

\[
\hat{p}_i = \mu + C(C + \sigma^2 I)^{-1}(q_i - \mu)
\]
Video denoising framework

Extract $n$ similar patches

Estimate model parameters

Wiener filtering

Aggregate by averaging

\[
\begin{bmatrix}
q_1, \ldots, q_n
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
\hat{p}_1, \ldots, \hat{p}_n
\end{bmatrix}
\]
Estimating the prior parameters

How to estimate $\mu$ and $C$?

**STEP 2:** Use the **guide** patches $g_1, \ldots, g_n \sim N(\mu, C)$:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} g_i$$

$$\hat{C} = \frac{1}{n} \sum_{i=1}^{n} (g_i - \hat{\mu}) (g_i - \hat{\mu})^T$$

**STEP 1:** Use the **noisy** pathes $q_1, \ldots, q_n \sim N(\mu, C + \sigma^2 I)$:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} q_i$$

$$\hat{C} \approx \hat{C}_q - \sigma^2 I = \frac{1}{n} \sum_{i=1}^{n} (q_i - \hat{\mu}) (q_i - \hat{\mu})^T - \sigma^2 I$$
Estimating the prior parameters

We use the following hard-thresholding of the eigenvalues of the sample covariance matrix of the noisy patches:

\[
\hat{\lambda}_i^H = H_\tau (\hat{\xi}_i - \sigma^2) = \begin{cases} 
\hat{\xi}_i - \sigma^2 & \text{if } \hat{\xi}_i \geq \tau \sigma^2 \\
0 & \text{if } \hat{\xi}_i < \tau \sigma^2.
\end{cases}
\]

Based on the results of Debashis Paul (2007) on the asymptotic properties for the eigenvalues and eigenvectors of the noisy sample covariance matrix.
Examples of local Gaussian models

Groups of 200 similar $9 \times 9 \times 4$ patches, in a $37 \times 37 \times 5$ search region.

Nearest neighbors 1 to 5.

Nearest neighbors 6 to 45.

Nearest neighbors 46 to 200.
VLNB: Examples of local Gaussian models

20 nearest neighbors (noisy).

Sample mean and first 19 principal directions.

Corresponding MAP estimates.
Patch search and aggregation

- 3D rectangular patches, typical sizes are $10 \times 10 \times 2$.

- Motion compensated search region: a square tracing the motion trajectory of the target patch. Motion estimation using optical flow (TV-L1 [Zach et al. '07]).

- Search region size: $21 \times 21 \times 9$

- Two iterations over the whole video (as in BM3D [Dabov et al. '07]).
Contents

Non-recursive methods

Recursive method I

Recursive method II

Empirical comparison
Motivation

\[ u_t = \text{recursive-method}(f_t, u_{t-1}) \]

Design constraints:
- reduce computational cost
- reduce memory cost \((M_t = O(1))\)

Appealing properties:
- natural mechanism to enforce temporal consistency
- can aggregate an arbitrary number of past frames
Motivation

\[ u_t, M_t = \text{recursive-method}(f_t, M_{t-1}) \]

Design constraints:

- reduce computational cost
- reduce memory cost \((M_t = O(1))\)

Appealing properties:

- natural mechanism to enforce temporal consistency
- can aggregate an arbitrary number of past frames
Dynamic Gaussian patch model

We assume a Gaussian linear dynamic model for a patch trajectory \( p_t \):

\[
\begin{align*}
    p_0 &= \mu_0 + w_0, & w_0 &\sim \mathcal{N}(0, P_0), \\
    p_t &= p_{t-1} + w_t, & w_t &\sim \mathcal{N}(0, W_t), \\
    q_t &= p_t + r_t, & r_t &\sim \mathcal{N}(0, \sigma^2 I).
\end{align*}
\]

"Static" model:

\[
\begin{align*}
    p &\sim \mathcal{N}(\mu, C), \\
    q &\sim \mathcal{N}(p, \sigma^2 I).
\end{align*}
\]
Dynamic Gaussian patch model

\[ p = p_0 + p_1 + p_2 + p_3 + p_4 \]

"Static" model:
\[
\begin{align*}
  p &\sim \mathcal{N}(\mu, C), \\
  q &\sim \mathcal{N}(p, \sigma^2 I).
\end{align*}
\]

We assume a Gaussian linear dynamic model for a patch trajectory \( p_t \):

\[
\begin{align*}
  p_0 &= \mu_0 + w_0, & w_0 &\sim \mathcal{N}(0, P_0), \\
  p_t &= p_{t-1} + w_t, & w_t &\sim \mathcal{N}(0, W_t), \\
  q_t &= p_t + r_t, & r_t &\sim \mathcal{N}(0, \sigma^2 I).
\end{align*}
\]
Dynamic Gaussian patch model

\[ p = p_0 \]

Full patch model:
\[ \begin{cases} 
  p \sim \mathcal{N}(\mu_K, Q_K^{-1}), \\
  q \sim \mathcal{N}(p, \sigma^2 I). 
\end{cases} \]

\[ \mu_K = \begin{pmatrix} 
  \mu_0 \\
  \mu_0 \\
  \mu_0 \\
  \mu_0 \\
  \mu_0 
\end{pmatrix}, \quad Q_K = \begin{pmatrix} 
  P_0^{-1} + W_1^{-1} & -W_1^{-1} \\
  -W_1^{-1} & W_1^{-1} + W_2^{-1} & -W_2^{-1} \\
  -W_2^{-1} & W_2^{-1} + W_3^{-1} & -W_3^{-1} \\
  -W_3^{-1} & W_3^{-1} + W_4^{-1} & -W_4^{-1} \\
  -W_4^{-1} & W_5^{-1} 
\end{pmatrix} \]
Recursive Bayesian patch estimation: filtering

\[
q = \begin{bmatrix}
q_0 \\
q_1 \\
q_2
\end{bmatrix}
\]

\[
\hat{p} = \begin{bmatrix}
\hat{p}_0 \\
\hat{p}_1 \\
\hat{p}_2
\end{bmatrix}
\]

\[
\hat{p}_2 = \mathbb{E}\{p_2|q_2, q_1, q_0\}
\]

**Kalman filter:** recursive computation of \( P(p_t|q_t, \ldots, q_1) \sim \mathcal{N}(\hat{p}_t, P_t) \)

\[
\begin{cases}
\hat{p}_t = (I - K_t)\hat{p}_{t-1} + K_t q_t & \text{state mean} \\
P_t = (I - K_t)(P_{t-1} + W_t)(I - K_t)^T + \sigma^2 K_t^2 & \text{state covariance} \\
K_t = (P_{t-1} + W_t)(P_{t-1} + W_t + \sigma^2 I)^{-1} & \text{Kalman gain}
\end{cases}
\]
Recursive Bayesian patch estimation: filtering

\[ q = \begin{array}{c}
\end{array} \]

\[ \hat{p} = \begin{array}{c}
\end{array} \]

\[ \hat{p}_2 = \mathcal{F}(q_2, \hat{p}_1, P_1, W_2) \}

Kalman filter: recursive computation of \( \mathbb{P}(p_t|q_t, \ldots, q_1) \sim \mathcal{N} \left( \hat{p}_t, P_t \right) \)

\[
\begin{align*}
\hat{p}_t &= (I - K_t)\hat{p}_{t-1} + K_t q_t \\
& \quad \text{state mean} \\
P_t &= (I - K_t)(P_{t-1} + W_t)(I - K_t)^T + \sigma^2 K_t^2 \\
& \quad \text{state covariance} \\
K_t &= (P_{t-1} + W_t)(P_{t-1} + W_t + \sigma^2 I)^{-1} \\
& \quad \text{Kalman gain}
\end{align*}
\]
Recursive Bayesian patch estimation: filtering

\[ q = \]

\[ \hat{p} = \]

Kalman filter: recursive computation of \( \mathbb{P}(p_t|q_t, \ldots, q_1) \sim \mathcal{N}(\hat{p}_t, P_t) \)

\[
\begin{align*}
\hat{p}_t &= (I - K_t)\hat{p}_{t-1} + K_t q_t & \text{state mean} \\

P_t &= (I - K_t)(P_{t-1} + W_t)(I - K_t)^T + \sigma^2 K_t^2 & \text{state covariance} \\

K_t &= (P_{t-1} + W_t)(P_{t-1} + W_t + \sigma^2 I)^{-1} & \text{Kalman gain}
\end{align*}
\]
Recursive Bayesian patch estimation: filtering

\[ q = \begin{array}{cccccccccc}
  \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\end{array} \]

\[ \hat{p} = \begin{array}{cccccccccccc}
  \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc \\
\end{array} \]

\[ \hat{p}_t = \mathbb{E}\{p_t | q_t, ..., q_0\} \]

**Kalman filter:** recursive computation of \( p_t | q_t, \ldots, q_1 \) \( \sim \mathcal{N}(\hat{p}_t, P_t) \)

\[
\begin{align*}
\hat{p}_t &= (I - K_t)\hat{p}_{t-1} + K_t q_t \\
\text{state mean} \\

P_t &= (I - K_t)(P_{t-1} + W_t)(I - K_t)^T + \sigma^2 K_t^2 \\
\text{state covariance} \\

K_t &= (P_{t-1} + W_t)(P_{t-1} + W_t + \sigma^2 I)^{-1} \\
\text{Kalman gain}
\end{align*}
\]
Recursive Bayesian patch estimation: smoothing

\[ q = \begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\end{array} \]

\[ \hat{p} = \begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\end{array} \]

\[ \hat{p}_t = \mathbb{E}\{p_t|q_T, \ldots, q_0\} \]

Rauch-Tung-Streibel smoother: back-recursion for \( P(p_t|q_T, \ldots, q_1) \sim \mathcal{N}(\tilde{p}_t, \tilde{P}_t) \)

\[
\begin{align*}
\tilde{p}_t &= (I - S_t)\hat{p}_t + S_t\tilde{p}_{t+1} & \text{state mean} \\
\tilde{P}_t &= P_t + S_t(\tilde{P}_{t+1} - P_t - W_{t+1})S_t & \text{state covariance} \\
S_t &= P_t(P_t + W_{t+1})^{-1} & \text{smoothing gain}
\end{align*}
\]
Parameter estimation

The only model parameters that need to be estimated are the state transition covariances \( W_t \) associated to the group.

\[
\mathbb{E}\{(q_t - q_{t-1})(q_t - q_{t-1})^T\} = W_t + 2\sigma^2 I.
\]

We assume that similar patches are iid realizations of the same dynamic model:

\[
\begin{align*}
\mathbf{p}_{t,i} &= \mathbf{p}_{t-1,i} + \mathbf{w}_{t,i} \text{ with } \mathbf{w}_{t,i} \sim \mathcal{N}(\mathbf{0}, W_t) \quad (1) \\
\mathbf{q}_{t,i} &= \mathbf{p}_{t,i} + \mathbf{r}_{t,i} \text{ with } \mathbf{r}_{t,i} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I). \quad (2)
\end{align*}
\]

We estimate \( W_t \) via the sample covariance matrix of the innovations:

\[
\hat{W}_t = \beta \hat{W}_{t-1} + (1 - \beta) \left( \frac{1}{n} \sum_{i=1}^{n} (\mathbf{q}_{t,i} - \mathbf{q}_{t-1,i})(\mathbf{q}_{t,i} - \mathbf{q}_{t-1,i})^T - 2\sigma^2 I \right) +
\]

A forgetting factor \( \beta \in [0, 1] \) is introduced to increase temporal stability.
Forward NL-Kalman filter: frame 1

1. Extract $n$ similar patches
2. Estimate model parameters
3. Wiener filtering
4. Aggregate by averaging
Forward NL-Kalman filter: frame 1

Extract $n$ similar patches

Wiener filtering

Aggregate by averaging

Estimate model parameters

Patch groups 1
Forward NL-Kalman filter: frame $t$

- Extract $n$ similar patches
- Estimates model parameters
- Coordinates $t-1$
- Kalman filtering
- Aggregate by averaging
- Patch groups $t-1$
- Patch groups $t$
Forward NL-Kalman filter: frame $t$

1. Extract $n$ similar patches
2. Estimate model parameters
3. Aggregation by averaging
4. Kalman filtering
Forward NL-Kalman filter: frame $t$

- Extract $n$ similar patches
- Estimate model parameters
- Aggregate by averaging

Patch groups $t - 1$ → $t$

- FOF $t - 1 \rightarrow t$
Forward NL-Kalman filter: frame $t$

1. Extract $n$ similar patches
2. Kalman filtering
3. Aggregate by averaging
4. Estimate model parameters
Forward NL-Kalman filter: frame $t$

- Extract $n$ similar patches
  - FOF $t-1 \rightarrow t$
  - Coordinates $t-1$
  - $\{q_{t,i}\}$
  - $\{q_{t-1,i}\}$
  - Estimate model parameters
  - $\{\hat{p}_{t-1,i}\}, P_{t-1}$

- Kalman filtering
  - $W_t$
  - $W_{t-1}$
  - $\{\hat{p}_{t,i}\}, P_t$

- Aggregate by averaging

- Patch groups $t-1$
  - Patch groups $t$
Forward NL-Kalman filter: frame $t$

1. Extract $n$ similar patches
2. Kalman filtering
3. Aggregate by averaging
4. Estimate model parameters

Patch groups $t - 1$

$\{q_{t,i}\}$

$\{q_{t-1,i}\}$

$\{p_{t-1,i}\}, P_{t-1}$

$\{\hat{p}_{t-1,i}\}, P_t$

$\{\hat{p}_{t,i}\}$
FNLK: Managing patch trajectories

- **Occlusions:** Terminate patch trajectories if an occlusion is detected.
- **Dis-occlusions:** Create groups for parts not covered by existing groups.

For each of these groups, we store

- coordinates of the patches in the group
- estimated clean patches $\hat{p}_{t,i}$
- covariance of estimated patches $P_t$
- transition covariance matrix $\hat{W}_t$
Contents

Non-recursive methods

Recursive method I

Recursive method II

Empirical comparison
Dropping the patch groups memory in FNLK

- Extract \( n \) similar patches
- Kalman filtering
- Aggregate by averaging

\[
\begin{align*}
\text{FOF} & \quad t - 1 \rightarrow t \\
\text{extract} & \quad \{q_{t,i}\} \\
\text{coordinates} & \quad \{\hat{q}_{t,i}\}, \{\hat{q}_{t-1,i}\} \\
\text{Estimate model parameters} & \\
\text{Patch groups} & \quad \{\hat{p}_{t-1,i}\}, P_{t-1} \\
\text{Patch groups} & \\
\end{align*}
\]
Dropping the patch groups memory in FNLK

| Patch groups $t-1$ | $f_{t-1}$ | $f_t$ | $u_t$ | $\{\hat{q}_{t,i}\}$ | $\{\hat{p}_{t-1,i}\}, P_{t-1}$ | $\{\hat{p}_{t,i}\}$ | $\{q_{t,i}\}$ | Aggregate by averaging |
|-------------------|--------|-------|-------|--------------------|-----------------|----------------|-----------------|---------------------|----------------------|

Extract $n$ similar patches

Kalman filtering

Estimate model parameters

FOF $t-1 \rightarrow t$
Recursive Bayesian patch estimation w/out covariances

Given \( \hat{p}_{t-1}, P_{t-1} \), at time \( t \) we have

\[
\begin{align*}
    p_{t-1} &\sim \mathcal{N}(\hat{p}_{t-1}, P_{t-1}), \\
p_t &= p_{t-1} + w_t, \quad w_t \sim \mathcal{N}(0, W_t), \\
q_t &= p_t + r_t \quad r_t \sim \mathcal{N}(0, \sigma^2 I).
\end{align*}
\]

Kalman filter: Recursive computation of \( \mathbb{P}(p_t | q_t, \ldots, q_1) \sim \mathcal{N}(\hat{p}_t, P_t) \)

\[
\begin{align*}
    \hat{p}_t &= (I - K_t)\hat{p}_{t-1} + K_t q_t, \quad \text{state mean} \\
    P_t &= (I - K_t)(P_{t-1} + W_t)(I - K_t)^T + \sigma^2 K_t^2, \quad \text{state covariance} \\
    K_t &= (P_{t-1} + W_t)(P_{t-1} + W_t + \sigma^2 I)^{-1}, \quad \text{Kalman gain}
\end{align*}
\]
Recursive Bayesian patch estimation w/out covariances

If we don’t have $\hat{p}_{t-1}, P_{t-1}$ we introduce them as parameters

$$
\begin{align*}
    p_{t-1} &\sim \mathcal{N}(\mu_{t-1}, C_{t-1}), \\
    p_t &= p_{t-1} + w_t, \quad w_t \sim \mathcal{N}(0, W_t), \\
    q_t &= p_t + r_t, \quad r_t \sim \mathcal{N}(0, \sigma^2 I).
\end{align*}
$$

We have the following posterior $\mathbb{P}(p_t | q_t) \sim \mathcal{N}(\hat{p}_t, P_t)$

$$
\begin{align*}
    \hat{p}_t &= (I - K_t)\mu_{t-1} + K_t q_t \quad \text{state mean} \\
    P_t &= (I - K_t)(C_{t-1} + W_t)(I - K_t)^T + \sigma^2 K_t^2 \quad \text{state covariance} \\
    K_t &= (C_{t-1} + W_t)(C_{t-1} + W_t + \sigma^2 I)^{-1} \quad \text{"Kalman" gain}
\end{align*}
$$
Parameter estimation for the memoryless model

We assume that similar patches are iid realizations of the same dynamic model:

\[
\begin{align*}
    p_{t-1,i} &\sim \mathcal{N}(\mu_{t-1}, C_{t-1}) \\
    p_{t,i} &= p_{t-1,i} + w_{t,i} \quad \quad w_{t,i} \sim \mathcal{N}(0, W_t) \\
    q_{t,i} &= p_{t,i} + r_{t,i} \quad \quad r_{t,i} \sim \mathcal{N}(0, \sigma^2 I).
\end{align*}
\]

\[
\begin{align*}
    \hat{\mu}_{t-1} &= \frac{1}{m} \sum_{i=1}^{m} p_{t-1,i} \\
    \hat{C}_{t-1} &= \frac{1}{m} \sum_{i=1}^{m} (p_{t-1,i} - \hat{\mu}_{t-1,i})(p_{t-1,i} - \hat{\mu}_{t-1,i})^T \\
    \hat{W}_t &= \frac{1}{n} \sum_{i=1}^{n} (q_{t,i} - p_{t-1,i})(q_{t,i} - p_{t-1,i})^T - \sigma^2.
\end{align*}
\]

**NOTE:** We introduced \( m \) to control the spatial averaging in \( \hat{\mu}_{t-1} \). Typically \( m < n \).
Additional simplification: work in DCT domain

**Assumption:** \( W_t = U \text{Diag}(\nu_t) U^T \) where \( U \) is the DCT basis

Then:

1. \( P_t = U \text{Diag}(\hat{\nu}_t) U^T \)
2. The Kalman recursion separates in \( d \) scalar filters on each DCT component:
Backward NL-Kalman filter: frame $t$

1. Extract $n$ similar patches
2. Kalman filtering
3. Aggregate by averaging

- Estimate model parameters
- $u_{t-1}/f_{t-1}$
- $f_t$
- $u_t$
Backward NL-Kalman filter: frame $t$

Extract $n$ similar patches

Kalman filtering

Aggregate by averaging

Estimate model parameters

$u_{t-1}/f_{t-1}$ $f_t$ $u_t$
Backward NL-Kalman filter: frame $t$

- Extract $n$ similar patches
- Kalman filtering
- Aggregate by averaging

- Estimate model parameters

$u_{t-1}^{w}/f_{t-1}^{w}$

$u_{t}$
Backward NL-Kalman filtering and smoothing

Algorithm 1: Recursive video filtering

**input**: Noisy video $f$, noise level $\sigma$

**output**: Denoised video $u$

1. for $t = 1 \ldots T$
   2. $v^b_t \leftarrow \text{compute-optical-flow}(f_t, \hat{u}_{t-1}, \sigma)$
   3. $\hat{u}^w_{t-1} \leftarrow \text{warp-bicubic}(\hat{u}_{t-1}, v^b_t)$
   4. $\hat{g}_t \leftarrow \text{bwd-nlkalman-filter}(f_t, \hat{u}^w_{t-1}, \sigma)$
   5. $\hat{u}_t \leftarrow \text{bwd-nlkalman-filter}(f_t, \hat{u}^w_{t-1}, \hat{g}_t, \sigma)$

Algorithm 2: Recursive video smoothing

**input**: Noisy video $f$, noise level $\sigma$

**output**: Denoised video $u$

1. for $t = 1 \ldots T$
   2. $v^f_t \leftarrow \text{compute-optical-flow}(\hat{u}_t, \tilde{u}_{t+1}, \sigma)$
   3. $\tilde{u}^w_{t+1} \leftarrow \text{warp-bicubic}(\tilde{u}_{t+1}, v^b_t)$
   4. $\hat{u}_t \leftarrow \text{bwd-nlkalman-smoother}(\hat{u}_t, \tilde{u}^w_{t-1}, \sigma)$
Three approaches based on Gaussian models of patches

**VNLB**
- Fixed 3D patch size
- No distinction between space and time
- Does not require OF
- Two iterations

**BNLK**
- Patch with arbitrary duration (kind of)
- Very cheap in memory
- Processing by raster order
- DCT domain (for the moment)
- Two iterations
- Simple smoother
- Sensitive to OF

**FNLK**
- Patch with arbitrary duration
- Processing organized by patch groups
- A lot of house-keeping
- Sensitive to OF
- Costly in memory
Contents

Non-recursive methods

Recursive method I

Recursive method II

Empirical comparison
### Quantitative denoising results (PSNR)

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR (σ=10)</th>
<th>PSNR (σ=20)</th>
<th>PSNR (σ=40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNLB</td>
<td>32.96</td>
<td>36.81</td>
<td>40.46</td>
</tr>
<tr>
<td>VNLnet</td>
<td>32.51</td>
<td>36.47</td>
<td>40.21</td>
</tr>
<tr>
<td>SPTWO</td>
<td>30.93</td>
<td>35.99</td>
<td>39.56</td>
</tr>
<tr>
<td>BM4D-OF</td>
<td>32.81</td>
<td>36.19</td>
<td>39.54</td>
</tr>
<tr>
<td>V-BM4D-mp</td>
<td>31.40</td>
<td>35.10</td>
<td>38.89</td>
</tr>
<tr>
<td>V-BM3D-np</td>
<td>31.11</td>
<td>34.68</td>
<td>38.24</td>
</tr>
<tr>
<td>BNLK2+S</td>
<td>31.95</td>
<td>35.26</td>
<td>38.69</td>
</tr>
<tr>
<td>BNLK2</td>
<td>31.19</td>
<td>34.59</td>
<td>38.17</td>
</tr>
<tr>
<td>BNLK1</td>
<td>28.20</td>
<td>32.19</td>
<td>36.20</td>
</tr>
<tr>
<td>FNLK</td>
<td>30.28</td>
<td>34.59</td>
<td>38.27</td>
</tr>
</tbody>
</table>

Average PSNR over 7 grayscale sequences $960 \times 540 \times 100$.  

- Red $\sigma = 10$  
- Orange $\sigma = 20$  
- Light red $\sigma = 40$
### Quantitative denoising results (SSIM)

<table>
<thead>
<tr>
<th>Method</th>
<th>SSIM (σ=10)</th>
<th>SSIM (σ=20)</th>
<th>SSIM (σ=40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNLB</td>
<td>0.8856</td>
<td>0.9428</td>
<td>0.9731</td>
</tr>
<tr>
<td>VNLnet</td>
<td>0.8752</td>
<td>0.9414</td>
<td>0.9732</td>
</tr>
<tr>
<td>SPTWO</td>
<td>0.7901</td>
<td>0.9368</td>
<td>0.9675</td>
</tr>
<tr>
<td>BM4D-OF</td>
<td>0.8836</td>
<td>0.9371</td>
<td>0.9671</td>
</tr>
<tr>
<td>V-BM4D-mp</td>
<td>0.8432</td>
<td>0.9169</td>
<td>0.9534</td>
</tr>
<tr>
<td>V-BM3D-mp</td>
<td>0.8360</td>
<td>0.9100</td>
<td>0.9599</td>
</tr>
<tr>
<td>BNLK2+S</td>
<td>0.8648</td>
<td>0.9251</td>
<td>0.9609</td>
</tr>
<tr>
<td>BNLK2</td>
<td>0.8460</td>
<td>0.9163</td>
<td>0.9570</td>
</tr>
<tr>
<td>BNLK1</td>
<td>0.8447</td>
<td>0.9278</td>
<td></td>
</tr>
<tr>
<td>FNLK</td>
<td>0.8245</td>
<td>0.9164</td>
<td>0.9575</td>
</tr>
</tbody>
</table>

Average SSIM over 7 grayscale sequences $960 \times 540 \times 100$.  

- Red: $\sigma = 10$  
- Orange: $\sigma = 20$  
- Pink: $\sigma = 40$
Conclusions and future work

▶ Current state-of-the-art in video denoising: either good results or fast results.

▶ Presented two recursive approaches bridging the gap between costly good methods and fast methods.

▶ They integrate information across longer time ranges and are allowed to recover many more details.

▶ Still very sensitive to optical flow.

Ongoing work

▶ Joint denoising and optical flow computation.

▶ Implement BNLK in an adaptive basis (instead of DCT).

▶ Fixed lag smoothers.

▶ Multiscale versions.
Thank you!

Reproducibility! Code and results:

- Non-recursive results:
  http://dev.ipol.im/~pariasm/video_denoising_models/

- VNLB: http://github.com/pariasm/vnlb

- SPTWO: https://doi.org/10.5201/ipol.2018.224

- BM4D-OF: https://github.com/pariasm/vbm3d

- BNLK: http://github.com/pariasm/bwd-nlkalman

- FNLK: http://github.com/tehret/nlkalman
References I


References II


Empirical Wiener filters in high dimensions

ML estimator of the inverse covariance matrix $Q_y = C_y^{-1}$ results from the following convex SDP:

$$\begin{align*}
\max & \quad \log \det Q_y - \text{tr}(Q_y \hat{C}_y) \\
\text{subject to} & \quad 0 < Q_y \preceq \sigma^{-2} I_d.
\end{align*}$$

Although there are efficient algorithms for solving such an SDP, they are still prohibitive for the present application.
Convergence of the sample covariance matrix

**Spiked covariance model:** Samples $x_1, \ldots, x_n \sim \mathcal{N}(0, C)$. We observe $y_i \sim \mathcal{N}(x_i, \sigma^2 I)$.

$$C = U \text{Diag}(\lambda_1, \ldots, \lambda_m, 0, \ldots, 0) U^T.$$  

**Theorem (Paul 2007).** Suppose that $d/n \to \gamma \in (0, 1)$ as $n \to \infty$. Let $\hat{\xi}_i$ be the $i$th eigenvector of the sample covariance matrix $\hat{C}_y$. Then $\hat{\xi}_i$ converges almost surely to

$$\hat{\xi}_i \to \begin{cases} \sigma^2 (1 + \sqrt{\gamma})^2 & \text{if } \lambda_i \leq \sqrt{\gamma} \sigma^2, \\ f(\lambda_i) := (\lambda_i + \sigma^2) \left(1 + \frac{\gamma \sigma^2}{\lambda_i}\right) & \text{if } \lambda_i > \sqrt{\gamma} \sigma^2. \end{cases}$$

We can estimate $\lambda_i$ as $\hat{\lambda}_i^S(\hat{\xi}_i) = \begin{cases} 0 & \text{if } \hat{\xi}_i \leq \sigma^2 (1 + \sqrt{\gamma})^2, \\ f^{-1}(\hat{\xi}_i) & \text{if } \hat{\xi}_i > \sigma^2 (1 + \sqrt{\gamma})^2. \end{cases}$
Convergence of the sample covariance matrix

Spiked covariance model: Samples $x_1, \ldots, x_n \sim \mathcal{N}(0, C)$. We observe $y_i \sim \mathcal{N}(x_i, \sigma^2 I)$.

$$C = U \operatorname{Diag}(\lambda_1, \ldots, \lambda_m, 0, \ldots, 0) U^T.$$

Theorem (Paul 2007). Suppose that $d/n \rightarrow \gamma \in (0, 1)$ as $n \rightarrow \infty$. Let $\hat{\xi}_i$ be the $i$th eigenvector of the sample covariance matrix $\hat{C}_y$. Then $\hat{\xi}_i$ converges almost surely to

$$\hat{\xi}_i \rightarrow \begin{cases} \sigma^2 (1 + \sqrt{\gamma})^2 & \text{if } \lambda_i \leq \sqrt{\gamma} \sigma^2, \\ f(\lambda_i) := (\lambda_i + \sigma^2) \left(1 + \frac{\gamma \sigma^2}{\lambda_i}\right) & \text{if } \lambda_i > \sqrt{\gamma} \sigma^2. \end{cases}$$

We can estimate $\lambda_i$ as

$$\hat{\lambda}_i^S(\hat{\xi}_i) = \begin{cases} 0 & \text{if } \hat{\xi}_i \leq \sigma^2 (1 + \sqrt{\gamma})^2, \\ f^{-1}(\hat{\xi}_i) & \text{if } \hat{\xi}_i > \sigma^2 (1 + \sqrt{\gamma})^2. \end{cases}$$
Estimators for the eigenvalues of $C_x$

$$\hat{\lambda}_S(\hat{\xi})$$

$$\hat{\xi} - \sigma^2(1 + \gamma)$$
\[ \hat{\lambda} = \lambda \hat{\lambda} = \hat{\xi} - \sigma^2 \]

\[ \hat{\lambda} = \hat{\lambda} S(\hat{\xi}) \sqrt{\gamma \sigma^2} \]
Variance threshold

We propose an additional estimator by hard thresholding the difference $\hat{\xi} - \sigma^2$. The value of the threshold is given by a parameter $\tau$:

$$
\hat{\lambda}_i^H = H_\tau(\hat{\xi}_i - \sigma^2) = \begin{cases} 
\hat{\xi} - \sigma^2 & \text{if } \hat{\xi} \geq \tau \sigma^2 \\
0 & \text{if } \hat{\xi} < \tau \sigma^2.
\end{cases}
$$
Denoising performance of both estimators

\[ \hat{\lambda} = \lambda \]
\[ \hat{\lambda} = \xi - \sigma^2 \]
\[ \hat{\lambda} = \hat{\lambda}_S(\hat{\xi}) \]
\[ \hat{\lambda} = H_\tau(\xi - \sigma^2) \]

Normalized MSE obtained by estimating the data samples from their noisy observations using empirical Wiener filters with different estimators for the \( a \) priori variances \( \hat{\lambda} \).