NON-LOCAL KALMAN: A RECURSIVE VIDEO DENOISING ALGORITHM

Thibaud Ehret, Jean-Michel Morel and Pablo Arias

CMLA, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France

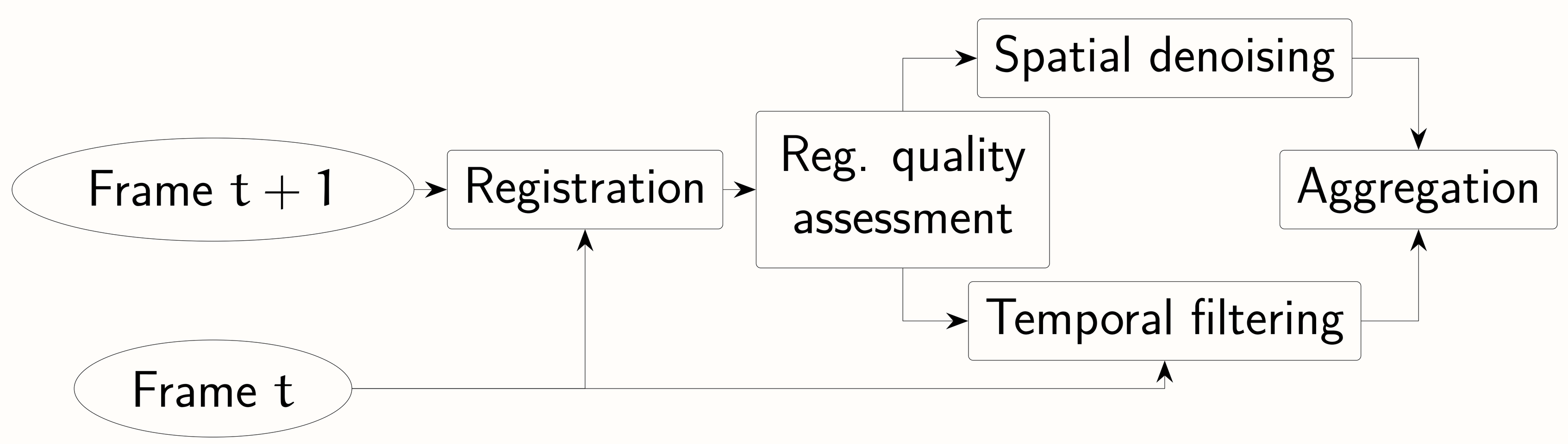
Contributions

We propose a recursive patch-based video denoising method with the following advantages:

- Particularly suited for real-time processing
- Produces temporally consistent videos, which are much better visually
- Competitive with current state-of-the-art denoising methods

Non-local Kalman

Pipeline of the recursive video denoising algorithm:



Registration and registration quality assessment

The registration is used to track patches from one frame to the next:

- Assumption: patches move with the optical flow of their center
- We use TV-L1 [1] on a downscaled version of the images to limit problems due to the noise

The detection of occlusion and missmatch can be done using a statistical framework:

- When a patch is correctly tracked by the optical flow the distance between the two instances of the patches follows a χ^2 distribution.
- The number of false alarm frame work based on ***a contrario*** gives an optimum threshold on the distance to detect occlusions and missmatch

Spatial denoising: creation of groups of trajectories

Use NL-Bayes [2] to create groups of patches and denoise them:

- Search for local nearest neighbors q_i for a given query patch q, these patches constitute the group
- Use a MAP estimate: $\hat{p} = \mu + C(C+\sigma^2 I)^{-1}(q-\mu)$ where $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} q_i$ and $\hat{C} = \frac{1}{N}\sum_{i=1}^{N} \overline{q}_i\overline{q}_i^T - \sigma^2 I$,

Temporal filtering: Kalman filtering along patches trajectories

For the patches in a group we assume the following model:

$$p_{t+1,i} = p_{t,i} + w_{t,i} \text{ with } w_{t,i} \sim \mathcal{N}(0, C_t)$$
$$q_{t,i} = p_{t,i} + n_{t,i} \text{ with } n_{t,i} \sim \mathcal{N}(0, \sigma^2 I).$$

where q_is are the observation (noisy patch) and the p_is what we tried to estimate. We apply Kalman filter [3] to estimate the $p_{t,i}$s.

Remarks

The study of the algorithm showed some limits on how video denoising algorithms are studied:

- Optical flow, while being suited to video processing, is difficult to use when the video has been degraded:
- PSNR are not a good fit to measure the quality of video denoising. The eye prefers a temporally consistent and it's not a problem if there's slight deformation.

References

[1] Zach et al. "A duality based approach for realtime TV-L 1 optical flow". *Joint Pattern Recognition Symposium*, 2007.

[2] Lebrun et al. "A nonlocal bayesian image denoising algorithm". *SIAM*, 2013.

[3] Kalman "A new approach to linear filtering and prediction problems." *Journal of basic Engineering 82*, 1960

[4] Dabov et al. "Video denoising by sparse 3D transform-domain collaborative filtering". *EUSIPCO*, 2007.

[5] Maggioni et al. "Video denoising by sparse 3D transform-domain collaborative filtering". *IEEE TIP*, 2012.

Experiments

Quantitative evaluation

Our algorithm is compared against VBM3D [4] and VBM4D [5], two state-of-the-art algorithms:

σ	Method	Bus	Foreman	Pedestrian_area	Crowd_run	Touchdown_pass	Station2	Average
10	VBM3D	33.32/.7824	**37.40**/.6681	**40.78**/.6577	35.62/.8017	39.08/.6103	38.92/.7266	37.52/.7078
	VBM4D	**33.39**/.8237	37.39/**.6871**	40.56/**.7463**	**35.69**/**.8457**	**39.60**/.6752	**39.93**/.7746	**37.76**/.7588
	NL-Kalman	33.34/**.8502**	36.16/.6782	38.67/.7420	34.29/.8383	38.82/**.6940**	39.91/**.7916**	36.86/**.7657**
	NL-Kalman (oracle)	*33.87/.8713*	*36.93/.7230*	*39.23/.7592*	*34.64/.8514*	*39.58/.7433*	*40.50/.8059*	*37.46/.7923*
20	VBM3D	29.57/.6064	34.60/.5763	**36.93**/.5579	**32.22**/.7122	36.09/.4703	35.45/.5689	34.14/.5820
	VBM4D	29.55/.6856	**34.61**/**.6073**	36.75/**.6468**	32.07/.7439	**36.41**/.4795	36.23/.6395	**34.27**/.6338
	NL-Kalman	**29.58**/**.7291**	33.19/.5844	35.61/.6444	30.89/**.7478**	35.91/**.5181**	**36.81**/**.6868**	33.66/**.6518**
	NL-Kalman (oracle)	*30.43/.7752*	*34.18/.6301*	*36.45/.6738*	*31.44/.7746*	*36.99/.6135*	*37.46/.7116*	*34.49/.6965*
30	VBM3D	**27.59**/.4995	32.77/.5224	34.44/.4869	**30.14**/.6394	34.55/.3906	33.36/.4536	32.14/.4987
	VBM4D	27.53/.5988	**32.91**/**.5612**	**34.45**/**.5745**	29.95/.6704	**34.76**/.3801	34.14/.5420	**32.29**/.5545
	NL-Kalman	27.30/**.6327**	31.27/.5335	33.27/.5680	28.64/**.6708**	33.91/**.4034**	**34.73**/**.5986**	31.52/**.5678**
	NL-Kalman (oracle)	*28.48/.6993*	*32.50/.5802*	*34.43/.6102*	*29.44/.7078*	*35.20/.5186*	*35.46/.6338*	*32.59/.6250*

Qualitative evaluation

Detail of the Crowd_run sequence (noise standard deviation $\sigma = 30$):

NL-Kalman NL-Kalman (oracle) VBM3D Original

Detail of the Pedestrian_area sequence (noise standard deviation $\sigma = 30$):

NL-Kalman NL-Kalman (oracle) VBM3D Original