International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:5, 2008

Adaptive kernel filtering used in video processing

Rasmus Engholm, Eva B. Vedel Jensen and Henrik Karstoft

***Abstract*—**In this paper we present a noise reduction filter for video processing. It is based on the recently proposed two dimensional steering kernel, extended to three dimensions and further augmented to suit the spatial-temporal domain of video processing. Two alternative filters are proposed - the time symmetric kernel and the time asymmetric kernel. The first reduces the noise on single sequences, but to handle the problems at scene shift the asymmetric kernel is introduced. The performance of both are tested on simulated data and on a real video sequence together with the existing steering kernel. The proposed kernels improves the Rooted Mean Squared Error (RMSE) compared to the original steering kernel method on video material.

***Keywords*—**Adaptive image filtering, Noise reduction, Kernel Methods, Video Processing.

I. Introduction

KERNEL methods are non-parametric methods of noise reduction. In relation to video it is a strength that the methods are non-parametric since broadcasted material have various origins and qualities. Electronic interference and flaws in the transmission and the digital image acquisition add noise to the image/video. Some programs are sent digitally at high bit rates, others are sent as analog signals via cables. Some programs have been stored on magnetic tapes for years and thereby degraded and some are sent directly. It seems impossible to construct a single parametric model that can describe all the different types of material and noise. An additional benefit of the kernel methods are that they have applications within noise reduction as well as within interpolation.

The transition to much larger screens, together with the transition from Catode Ray Tube technology (CRT) to Liquid Crystal Displays (LCD) and similar digital screen technologies, makes noise appear much more disturbing on a television set. It is therefore important that the noise is reduced to get the viewer the best perceived experience possible. However other factors such as sharp edges are also important for the perceived video quality. This is what makes the noise removal difficult because blurring reduces the noise but also removes the sharpness of the edges. On the other hand edge enhancement techniques e.g. peaking enhances the noise. Another similar problem is that texture and noise shares many of the same statistics e.g. a sandy beach or the leafs of a tree can locally appear very random.

We will in section II first describe related work on kernel methods with special emphasis on the recently proposed steering kernel method. In section III we extend this to the video domain, and augment it to handle some of the difficulties in video. We present experiments on both video with simulated noise and real analog noise in section IV to compare and show the behavior of various methods considered. The application we have in mind is television but the scope of kernel methods is much broader.

Rasmus Engholm is a PhD candidate at the University of Aarhus and working at Bang & Olufsen a/s, Denmark, email: engholm@imf.au.dk

Eva B. Vedel Jensen is a Professor in statistics at the University of Aarhus, email: eva@imf.au.dk

Henrik Karstoft is an Associate Professor in signal processing at the Engineering College of Aarhus, email:hka@iha.dk

II. Related Work

In recent years methods that utilizes non-static kernels, dynamic kernels, that adapt to the data, have been proposed, see [1], [2]. These kernels add flexibility to the performance as well as introduce new challenges.

Instead of just using the Euclidean distance to determine the weight of the observation at a given site we may combine the Euclidean distance with other metrics. Examples of this, where the observed value also enters into the computation of the weight are bilateral filter and steering kernels.

A. Bilateral filter

The bilateral kernel has had a lot of support since the introduction in 1998 [2] due to its edge preserving properties combined with the appealing simplicity and intuitive construction. The bilateral filter is an attempt to build a filter kernel that has the smoothing properties of the ordinary Gauss filter, and at the same time does not smooth the edges.

The intuition is that sites with similar observations which are also located close to each other is from the same structure and therefore can be assigned larger weights than sites with either different observations or location far away from each other. The bilateral kernel is then a product of two kernels. In general any low pass kernel can be used, but usually the Gaussian kernel is used.

This intuitive approach has received a lot of theoretical investigation tying it together with other edge preserving filtering techniques such as anisotopic difussion, robust estimation, weighted least squares [3] and adaptive smoothing [4], [5]. Others have tried to speed up the performance [6]–[8]. The algorithm has also found use within computer graphics where it has been used to perform smoothing of 3D-meshes [9]. The origin of the bilateral filter is empirical, but it can be derived from the weighted least squares formulation, similar to 0-order local regression

On artificial examples the filter performs very satisfying. But on natural images the shortcomings becomes clear. The filtered images often appear 'cartoon' like. The filter does indeed sustain the sharp edges, but within the areas defined by the edges, all fine details get blurred. Another disadvantage is that a single noise observation (a pixel very different from the neighborhood is is located in) remains unchanged.

B. Steering kernels

Instead of using the difference in intensities to guide the filtering like the bilateral filter, thus indirectly using gradient information, the intuition behind the steering kernel, introduced in [1], is to directly use the gradient information to construct the kernel. This means that the actual kernel will be a deformation of an ordinary smoothing kernel (scaled, rotated, skewed). The steering kernel was introduced with a Gaussian kernel, but any type of kernel could in principle be used. We should note that the original formulation suggest that the kernel is used iteratively. However the optimal number of iterations remains unclear, why we here will only consider it as a one pass filter.

As with the bilateral filter, the objective is to remove noise while preserving the edges. The shortcomings of the bilateral filter described above appear partly because the kernel is designed directly from the observations i.e. a single black pixel in a white neighborhood can be hard to smooth, since the intensity part of the kernel will ensure that all weight will be put on the single observation under consideration.

The steering kernels are defined using the gradient information and performing principal component analysis, to reduce all the gradients in a local window to the dominant orientation in that window. A kernel is then calculated based on this orientation giving weights to the observations in the window such that most smoothing is performed along edges and not over edges.

We define the pixel that we want to estimate as $\mathbf{x_0}$ in a local window. The sites of the observations are indexed in a two dimensional grid $(x_{1i}, x_{2j}), i = 1, \ldots, n, j = 1 \ldots, m$, representing our pixel in the plane. To each pair (x_{1i}, x_{2j}) we have a single associated observation, y_{ij} giving the grey level value in the image.

We consider a local window and define the gradient as

$$(\nabla y)(x_{1i}, x_{2j}) = \left[\frac{\partial y}{\partial x_1}(x_{1i}, x_{2j}), \frac{\partial y}{\partial x_2}(x_{1i}, x_{2j})\right]$$

Since we are working on discrete values we must use a discrete differentiation operator.

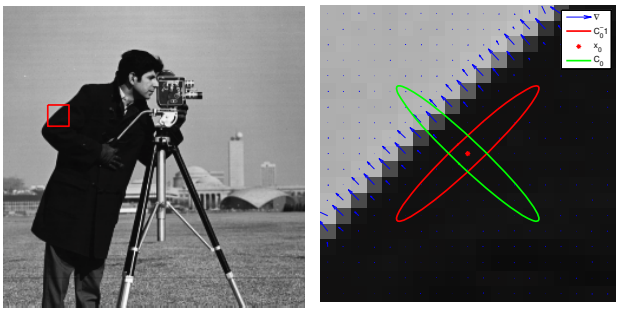
We are interested in finding the dominant orientation of the structure in the local window of the image, so that we do not smooth in that direction. On average we can consider the gradients to be orthogonal to the dominant orientation. We do a PCA analysis on the gradients in the local window to find the dominant edge orientation. The gradients of a local window is then reordered as an $nm \times 2$ matrix:

$$G = \begin{bmatrix} (\nabla y)(x_{11}, x_{21}) \\ (\nabla y)(x_{11}, x_{22}) \\ \vdots \\ (\nabla y)(x_{11}, x_{2m}) \\ (\nabla y)(x_{12}, x_{21}) \\ \vdots \\ (\nabla y)(x_{1n}, x_{2m}) \end{bmatrix},$$

where each row specifies first the gradient in the horizontal and then in the vertical direction. The steering kernel uses a simple estimate of the covariance structure of the gradients obtained by

$$\mathbf{C_{x_0}} = \frac{1}{nm}\sum_{i,j}\left[\begin{array}{cc} (\frac{\partial y}{\partial x_1})^2 & \frac{\partial y}{\partial x_1}\frac{\partial y}{\partial x_2} \\ \frac{\partial y}{\partial x_1}\frac{\partial y}{\partial x_2} & (\frac{\partial y}{\partial x_2})^2 \end{array}\right]\Bigg|_{(x_{1i}, x_{2j})} = \frac{1}{nm}G^T G$$

This symmetric structure, $\mathbf{C_{x_0}}$, of the gradients in a local window is then used to form the steering kernel. The objective is to spread the kernel parallel to the edges, so the edges remain unblurred. Secondly if the gradients in general are small we would assume that the differences originates from noise and need smoothing thus requiring a relatively flat kernel. On the other hand if the gradients in general are large we will assume that it originates from structure and need less smoothing which calls for a steep kernel. Considering the eigenvectors of $\mathbf{C_{x_0}}$, the first eigenvector is in the direction of greatest variance i.e. the direction of on average greatest gradient. This structure is exactly the opposite of what we look for. We wish to rotate the structure and make the largest eigenvalue orthogonal to the main gradient direction. This can be accomplished by inverting $\mathbf{C_{x_0}}$. Since the inverse of a symmetric matrix, has the same eigenvectors but reciprocal eigenvalues, we obtain the wanted structure, see Fig. 1.



(a) (b)

Fig. 1. (a) Image with a red square that indicates the neighborhood being examined. (b) Contour ellipses of $\mathbf{C_{x_0}}$ and $\mathbf{C_{x_0}}^{-1}$ superimposed onto the current neighborhood together with the estimated gradients.

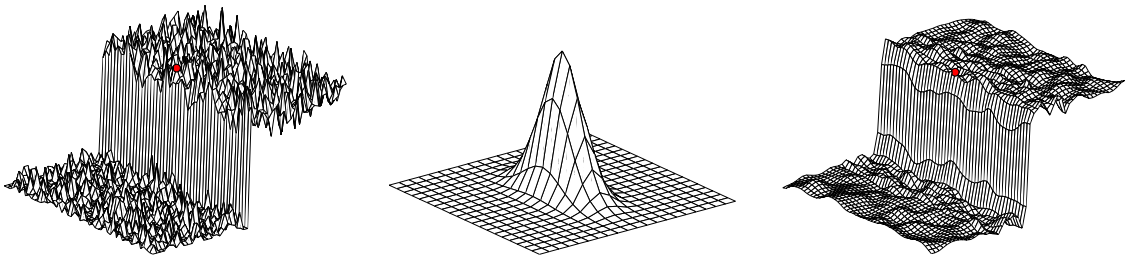
Using this structure to construct a Gaussian kernel we arrive at the steering kernel

$$K_S(\mathbf{x_0}, \mathbf{x_w}) = c \cdot \exp\left(-\frac{(\mathbf{x_0} - \mathbf{x_w})^{\mathbf{T}}(\mathbf{C_{x_0}})(\mathbf{x_0} - \mathbf{x_w})}{2\sigma_s}\right)$$

where $K_S(\mathbf{x_0}, \mathbf{x_w})$ is the weight associated to pixel $\mathbf{x_w}$ in the local window, $\mathbf{C_{x_0}}$ is the structure of the gradients for the local window centered around $\mathbf{x_0}$, and c is a normalizing constant. σ_s is the parameter governing the basic level of smoothing e.g. the steepness of the Gaussian kernel before it is deformed by C_0. Fig. 2 illustrates the construction of a steering kernel.

III. Steering Kernels in Video

Applying steering kernels on video is an obvious extension. By having more frames we want to use the redundancy of the video to get a better estimate of the pixel value.



(a) (b) (c)

Fig. 2. Construction of the steering kernel based on the red dot in (a) in a 21x21 neighborhood. (b) The steering kernel based on the structure of the gradients. (c) The result of filtering with the steering kernel with parameter $\sigma_s = 11$.

The redundancy occurs because in video, e.g. 50 frames per second, are available. Of course the type of content determine how useful the surrounding frames are. If a scene shift occurs we cannot use the information to get a better estimate. We propose firstly to extend the idea of the steering kernel such that the kernel is three dimensional and secondly to introduce time asymmetric kernels to handle the problems at scene shifts.

With grey level video we have a three dimensional site input $(x_{1i}, x_{2j}, x_{3k}), i = 1, \ldots, n, j = 1 \ldots, m, k = 1 \ldots, l$, where (x_{1i}, x_{2j}) indicates the spatial location of the site in the frame and x_{3k} the frame number, i.e. the temporal location, of the site. The observation at site (x_{1i}, x_{2j}, x_{3k}) is denoted y_{ijk} representing the grey level value.

A. Time symmetric kernel

Extending the steering kernel introduces some difficulties, since the temporal resolution can not be compared to the spatial resolution because of different scales. By governing the time direction by a separate parameter, we construct a video version of the steering kernel. We restrict the kernel such that two of the three axes in the estimated covariance structure is parallel to the spatial domain (the one from the original steering kernel). The third is then orthogonal to these and parallel to the temporal directional plane as shown on Fig. 3. This is chosen because we wish to maintain sharp edges with respect to each frame and not in arbitrarily directions in the spatial/temporal space.

We construct the matrix similar to the two dimensional case but augment the matrix to a 3×3 matrix. We reuse the $\mathbf{C_{x_0}}$ unchanged but augment it with the variance of the gradient in the temporal direction

It should be emphasized that we for the calculation of the temporal variance use all gradients in the temporal direction in the considered block, whereas we only use the gradients in the the spatial directions from the central frame. The resulting shape of the video kernel in the central frame is the same shape as in the 2 dimensional case, but with smaller weight since some of the weight is transferred to the surrounding frames

$$d_{\mathbf{x_0}} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l}\left(\frac{\partial y}{\partial x_3}(x_{1i}, x_{2j}, x_{3k})\right)^2.$$

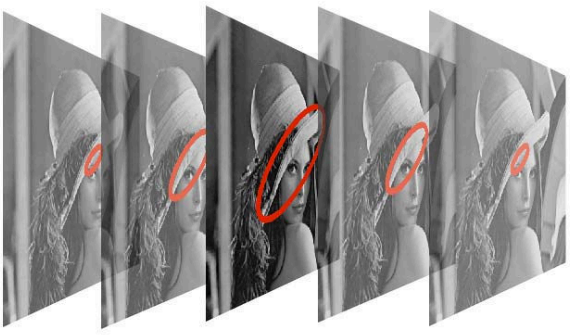


Fig. 3. Illustration of the idea of video steering kernel, here using five relatively identical frames. The third is the one we perform filtering on. The superimposed red ellipses represent level curves with same density of the kernel. It is obvious that redistributing some of the kernel weight on surrounding frames in this case can give a better estimate.

Here $d_{\mathbf{x_0}}$ is the last entry in the diagonal of our enlarged matrix.

We can now define the covariance structure of the time symmetric kernel as

$$\tilde{\mathbf{C}}_{\mathbf{x_0}} = \begin{bmatrix} \frac{\mathbf{C_{x_0}}}{\sigma_s} & 0 \\ 0 & \frac{d_{\mathbf{x_0}}}{\sigma_t} \end{bmatrix},$$

where $\mathbf{C_{x_0}}$ is exactly as before. Since we want to be able to adjust the overall smoothing in spatial and temporal direction independently we introduce a new parameter σ_t governing the temporal degree of smoothing. We can now define the time symmetric steering kernel as

$$K_{TS}(\mathbf{x_0}, \mathbf{x_w}) = c \cdot \exp\left(-\frac{(\mathbf{x_0} - \mathbf{x_w})^{\mathbf{T}}(\tilde{\mathbf{C}}_{\mathbf{x_0}})(\mathbf{x_0} - \mathbf{x_w})}{2}\right),$$

where w index the sites in the local block and c is the normalizing constant.

We see that the kernel is locked to have one eigenvector parallel to the temporal direction, since there is no correlation between the temporal and the spatial direction. Furthermore we notice that when experiencing small temporal gradients, i.e. the content of the frame is relatively unchanged, the estimate of $\tilde{\mathbf{C}}_{\mathbf{x_0}}$ is going to be relatively small. This will lead to a kernel

stretching in the temporal direction giving more weight to the surrounding frames. On the other hand if is there is a lot of changes, we will get a large estimate of $\tilde{\mathbf{C}}_{\mathbf{x_0}}$, resulting in a kernel shrinking in the temporal direction ultimately becoming the ordinary steering kernel.

B. Time asymmetric kernel

Utilizing the surrounding frames like the described kernel construction above improves noise reduction, however the performance is degraded at scene shift and at pan. Obviously at scene shifts the gradient in the temporal direction is relatively large, resulting in a very flat kernel in the temporal direction, almost identical to the two dimensional kernel. We therefore propose an alternative kernel construction, where the temporal direction supports asymmetry. This provides us with more flexibility to exploit the redundancy in only one temporal direction when entering/exiting a scene shift as illustrated in Fig. 4. For estimating the asymmetric kernel we have to

Fig. 4. Illustration of the idea of time asymmetric kernel, here using three relatively identical frames and a scene shift. The third is the one we perform filtering on. The superimposed red ellipses represent level curves with same density of the kernel. It is obvious that redistributing some of the kernel weight to frame 1 and 2 will give a better estimate, and that frame 4 and 5 should have little or zero weight

estimate the two parts of the kernel separately. We therefore divide the gradients into two groups, one containing the gradients based on frames before our current frame and one after the current frame. We then get

$$\tilde{\tilde{\mathbf{C}}}_{\mathbf{x_0}} = \begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{t-1}(\frac{\partial y}{\partial x_3}(x_{1i},x_{2j},x_{3k})_3)^2 \\ \quad \text{if } x_{3k} \leq t \\ \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=t+1}^{l}(\frac{\partial y}{\partial x_3}(x_{1i},x_{2j},x_{3k})_3)^2 \\ \quad \text{otherwise} \end{cases},$$

where t is the index of the current frame, x_{3k} is the index in the temporal direction in the considered block. $\tilde{\tilde{\mathbf{C}}}_{\mathbf{x_0}}$ substitutes the $\tilde{\mathbf{C}}_{\mathbf{x_0}}$ from the time symmetric kernel.

IV. Experiments

We tested the proposed kernels in four different setups illustrating the performance. We tested together with the regular two dimensional steering kernel for comparison. The first three we used sequences with added normal noise with N(0,5). The last test was done on a real broadcasted video with analog noise originating from transmission. For the simulated data we optimized wrt. the Rooted Mean Squared Error (RMSE). The last example we used subjective image quality.

A. Single static sequence

In the first experiment, we took a standard sequence, *Susie* representing respectively a very static setting. We added noise with $N(0,5)$. We compared the performance of the steering kernel of size [5 x 5]with the video steering kernel of size [5 $\times$ 5 $\times$ 3] with respect to RMSE. For parameter estimation we used a frame in the middle of the sequence to determine the steering kernel parameter. This was shared with the video steering kernels. we notice a significant improvement from the two dimensional kernel. The difference between the three dimensional kernels with and without symmetry is ambiguous. There is no clear winner. For this sequence, see Fig. 5, the perceptual quality appears much better for the video steering kernels than the original two dimensional steering kernel.

B. Single dynamic sequence

The setup of this experiment is identical to the previous. The used sequence, *Football*, represents a very dynamic sequence with both camera and object movement. Adding movement makes the video steering less optimal. However the performance is still constantly superior to the two dimensional kernel. The variance of the video kernels performance is more pronounced than in the previous. The time asymmetric kernel is slightly better than the time symmetric kernel.

C. Sequences with scene shifts

We used another video consisting of five very different concatenated standard sequences, see Fig. 7. Those were *Table Tennis*, a relative static scene with a noise like wall paper, *Flower Garden* a panoramic sequence of small flowers, *Birdcage*, a very static scene with a birdcage spinning slowly around, *Susie* as mentioned before and finally *Mobcal*, a panoramic video of mobile and calender. We added noise as before to investigate the performance when encountering scene shift. The three dimensional kernels performed better than the two dimensional kernels. Only exception is in the *Flower Garden* sequence where the performance of all three are similar bad. We notice that the biggest difference between the two dimensional and the three dimensional is obtained in sequences where the camera is relatively static and the background has a noise like texture e.g. *Table Tennis*. Comparing the time symmetric kernel and the time asymmetric kernel we notice that in general the asymmetric kernel performs better. Furthermore at scene shift indicated with a dashed vertical line on Fig. 7(f) the performance of the asymmetric kernel is superior to the orthogonal (most noticeable around frame number 90).

D. Real video sequence

The final test was performed on a real video which had been broadcasted and grabbed and converted to digital video. We only show a frame together with the residual image to illustrate the effect of the three different methods, since the original sequence is unavailable. Each of the RGB-layers have been filtered separately. The parameter was chosen to make the best visual filtering. On the filtered images we notice that the video kernels appear most noise free.

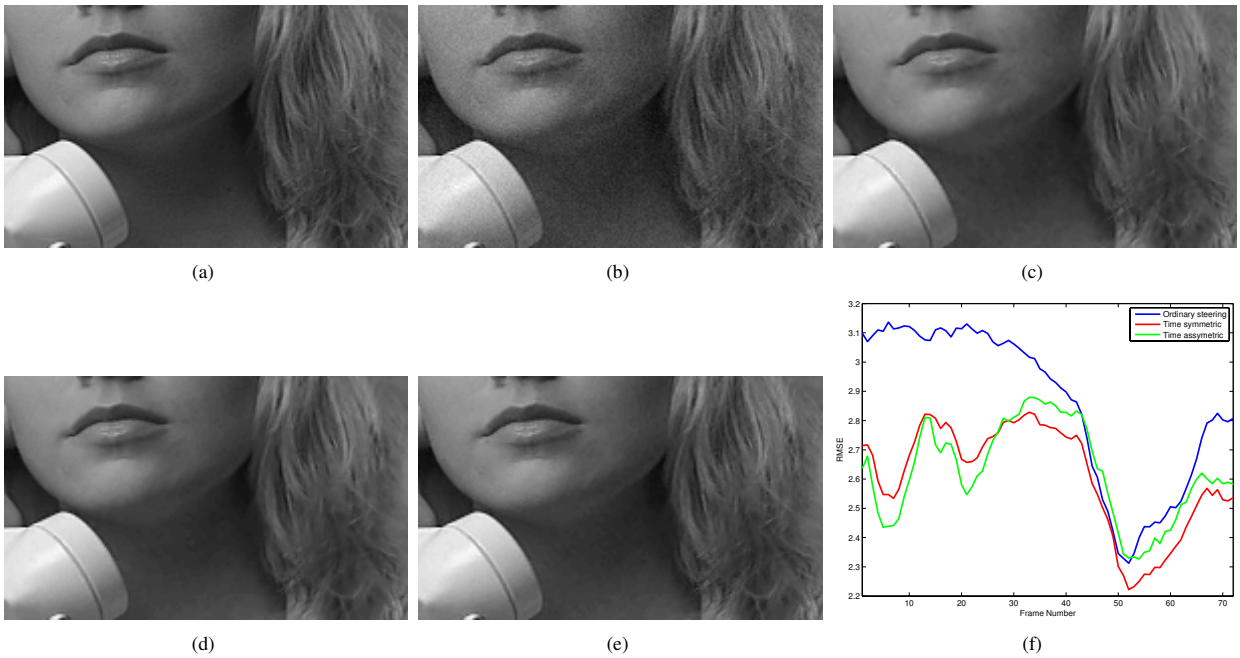


Fig. 5. Detail of frame number 7 of, (a) the original frame and, (b) the frame with added noise with $N(0, 5)$. The resulting frame of filtering with the, (c) steering kernel of size [5 × 5], (d) video steering kernel of size [5 × 5 × 3] and (e) video steering kernel of size [5 × 5 × 3] and (f) The frame by frame comparison of the RMSE of the three methods.

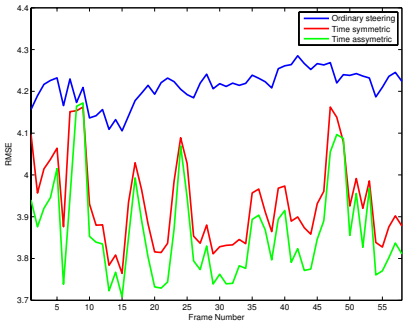


Fig. 6. The frame by frame comparison of the RMSE of the three methods on the *Football*. The kernel sizes was like the *Susie* example.

V. Conclusion

In this paper we have extended the steering kernel to be used in video with added flexibility. The kernel have been altered progressively to handle the difficulties concerning scene shift. This results in the introduction of a constraint on direction of the temporal direction and the introduction of asymmetric kernels. The proposed kernels improves the performance with respect to minimizing the RMSE compared to the original steering kernel. Further work with optimizing the choice of parameters needs to be performed. Finally further work also needs to be done to show how the obtained RMSE results correlates with the perceived video quality.

References

[1] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 349–366, 2007.

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *Proc. 6th Int. Conf. Computer Vision, New Delhi, India*, pp. 839–846, 1998.

[3] M. Elad, "On the origin of the bilateral filter and ways to improve it," *Image Processing, IEEE Transactions on*, vol. 11, no. 10, pp. 1141–1151, 2002.

[4] D. Barash, "Fundamental relationship between bilateral filtering, adaptivesmoothing, and the nonlinear diffusion equation," *PAMI, IEEE Transactions on*, vol. 24, no. 6, pp. 844–847, 2002.

[5] D. Barash and D. Comaniciu, "A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift," *Image and Vision Computing*, vol. 22, no. 1, pp. 73–81, 2004.

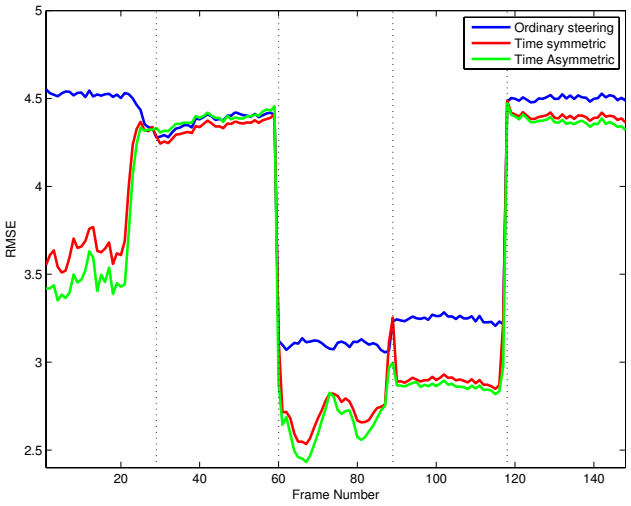
[6] T. Pham and L. van Vliet, "Separable Bilateral Filtering for Fast Video Preprocessing," *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pp. 454–457, 2005.

[7] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 257–266, 2002.

[8] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *Proc. of Eur. Conf. on Comp. Vision*, 2006.

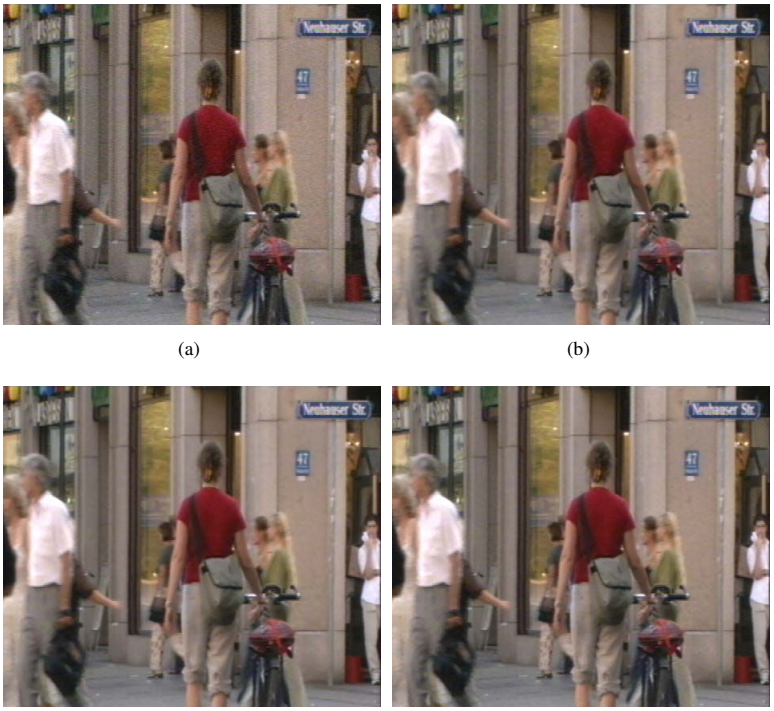
[9] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," *Int. Conf. on Comp. Graphics and Interactive Techniques*, pp. 950–953, 2003.

(a) (b) (c) (d) (e)



(f)

Fig. 7. (a)-(e) A single frame from each of the five subsequences in this sequence. The scene was added noise with $N(0, 5)$ and a filtering with the steering kernel and the video steering kernel was performed. The resulting RMSE of the filtering is presented in (f). The vertical lines marks, where a scene shift occurs. The video kernels outperforms the two dimensional kernel. The time asymmetric kernel performs overall best and handless scene shift better than the time symmetric kernel.



(a) (b)

(c) (d)

Fig. 8. (a) A frame from a real video sequence. (b), the result of the filtered frame with the two dimensional steering kernel, (c) the time symmetric kernel and (d) time asymmetric kernel.