SIMD implementation of the bilateral filter

Michael M. Bronstein
Department of Computer Science
Technion – Israel Institute of Technology
Haifa 32000, Israel
September 7, 2009

Abstract

Bilateral filter is a state-of-the-art method for noise reduction and color depth enhancement. The plausible visual result the filter produces makes it highly desired for video processing applications, yet, its high computational complexity makes a real-time implementation a challenging task. Presented here is a highly-parallel version of the bilateral filter for noise reduction in color video sequences, oriented to SIMD-type architecture.

Keywords: Video processing, noise reduction, bilateral filter, retinex.

1 Introduction

Noise reduction is an important part of any video processing pipeline. Here, we understand the term “noise” in a broad sense, referring to random noise present in analog video sources, digital video artifacts resulting from MPEG compression (high-frequency quantization and blocking), false contouring artifacts resulting
from quantization, and color artifacts resulting from chroma subsampling. Reducing such noise, in addition to improving the visual quality of the video, also allows improve compressibility of video sequences and facilitate block matching in motion-adaptive video processing algorithms.

1.1 Bilateral filter

Bilateral filter is a state-of-the-art denoising algorithm, referring to a class of method for non-linear, content adaptive image processing, introduced in the formulation of a PDE processing referred to as Beltrami flow in [1, 2] and, in different formulation (which can be considered as an approximation of the Beltrami flow) in [3, 4]. A byproduct of the bilateral filter is the color depth enhancement, an effect referred to as super-precision. Bilateral-type filters are often applied as the first stage in a video enhancement pipeline, followed by other processing and enhancement algorithms (deinterlacing, frame rate conversion, upscaling, etc.) In addition to direct applications related to noise reduction, bilateral filter is the core of many adaptive dynamic range extension (retinex-like) algorithms [5, 6].

Given a grayscale image $y_{k,j}, k = 1, ..., N; j = 1, ..., M$ (for example, $y_{k,j}$ can be the luma channel of a color image in the YUV color space) with intensity levels in the range $[0, 255]$, the filter computes each output pixel $\hat{y}_{k,j}$ as a weighted average of its neighbors in the window around $y_{k,j}$. The weights are inversely proportional to the Euclidean and radiometric distance between the pixels,

$$\hat{y}_{k,j} = \frac{1}{w_{k,j}} \sum_{m=-P_y}^{P_y} \sum_{n=-P_y}^{P_y} e^{-\frac{m^2+n^2}{2\sigma_s^2}} e^{-\frac{(y_{k,j} - y_{k,m,j-n})^2}{2\sigma_y^2}} y_{k,m,j-n},$$

(1)

where

$$w_{k,j} = \sum_{m=-P_y}^{P_y} \sum_{n=-P_y}^{P_y} e^{-\frac{m^2+n^2}{2\sigma_s^2}} e^{-\frac{(y_{k,j} - y_{k,m,j-n})^2}{2\sigma_y^2}}.$$
is a normalization guaranteeing that all the filter coefficients add up to one. $\sigma_s$ and $\sigma_y$ are the spatial and radiometric variance parameters governing the filter behavior (roughly, the larger the variance, the smoother is the result). The window size is $(2P_y + 1) \times (2P_y + 1)$ pixels, with $P_y$ typically varying between 5 to 15.

Straightforward computation of the bilateral filter is performed using a sliding window. For each pixel in raster scan order, neighbor pixels within a window around it are taken and used to compute the filter output; the window is moved right by one pixel and so on (Figure 2, top).

Due to the large amount of computations, a real-time implementation of the bilateral filter in full high-definition (HD) resolution is extremely challenging. Assuming 1080p30 video ($1920 \times 1080$, 30Hz), we have to compute $62.2 \times 10^6$ results per second. Several accelerations have been proposed. For the Beltrami flow, the linear one dimensional (LOD) scheme appears to be efficient [7, 8]. For the bilateral filter, Durand and Dorsey [9] and Paris and Durand [10] showed an approximation by a sum of liner filters. Pham and van Vliet [11] showed a separable version of the bilateral filter. Weiss [12] proposed a method based on an efficient histogram computation.

The fact that the bilateral filter applies the same processing at every pixel makes it especially suitable for SIMD (single instruction multiple data) architectures of digital signal processors. Section 2 describes the Durand-Dorsey acceleration. In Section 3, we extend the approach of Durand and Dorsey to color video sequences, proposing temporal averaging and coupling of the luma and chroma channels. In such a version, in addition to efficiently removing noise and blocking artifacts, our bilateral filter also improves the spatial resolution of the chroma channels, allowing to compensate for color bleeding resulting from 4:2:0 chroma subsampling. In Section 4 we describe such an efficient parallel implementation using the lazy sliding window approach. Section 5 shows some simulation results.
Section 6 concludes the paper.

2 Durand-Dorsey acceleration

If the intensity $y_{i,j}$ is assumed a constant $c$, we can express (1) as a linear filter,

$$\hat{y}_{k,j} = \frac{1}{w_{k,j}} \sum_{m=-P_y}^{P_y} \sum_{n=-P_y}^{P_y} e^{-\frac{m^2+n^2}{2\sigma_y^2}} e^{-\frac{(c-y_{k-m,j-n})^2}{2\sigma_y^2}} y_{k-m,j-n}$$

(2)

where

$$g_{i,j} = e^{-\frac{(c-y_{i,j})^2}{2\sigma_y^2}} y_{i,j},$$

is computed by applying a Gaussian non-linearity and $w_{k,j}$ are normalization factors defined by

$$w_{k,j} = \sum_{m=-P_y}^{P_y} \sum_{n=-P_y}^{P_y} e^{-\frac{m^2+n^2}{2\sigma_y^2}} e^{-\frac{(c-y_{k-m,j-n})^2}{2\sigma_y^2}}.$$

Using this observation, Durand and Dorsey [9] proposed an approximation to the bilateral filter by dividing the dynamic range of the image intensity levels into $L_y + 1$ equal bands using a linear filter approximation (3) in each band. The results are then merged in the following way:

$$\hat{y}_{i,j} \approx \sum_{l=0}^{L_y} \mu_{i,j} \hat{y}_{i,j}^{\Delta y},$$

where $\Delta_y = 255 L_y^{-1}$ is the intensity band width and $\mu_{i,j} = \max\{1 - \Delta_y^{-1} |y_{i,j} - l\Delta_y|, 0\}$ is a soft mask determining whether the $(i, j)$-th pixel belongs to the $l$-th band.

The Durand-Dorsey approach offers a significant acceleration compared to the straightforward bilateral filter computation. The number of intensity levels $L_y$
(typically, between 10 and 50) is a parameter for performance-quality tradeoff. Moreover, since the spatial variance $\sigma_s$ is usually relatively large compared to the window size, the spatial part of the filter can be approximated by a constant, i.e., Gaussian blur can be replaced by simple averaging, which is computed using sliding window [12]. A further acceleration can be achieved by first reducing the resolution of $y_{i,j}$ and then performing the averaging; in this case, a smaller window is needed. The merging is performed after upscaling the results to the original resolution.

3 Extensions

3.1 Spatio-temporal filtering

Similarly to the Beltrami flow, the bilateral filter can be straightforwardly extended to averaging not only in space (on adjacent pixels), but also in time (on subsequent frames),

$$
\hat{y}_{k,j,t} = \frac{1}{w_{k,j,t}} \sum_{\tau=-T}^{T} \sum_{m=-P_y}^{P_y} \sum_{n=-P_y}^{P_y} e^{-\frac{m^2+n^2}{2\sigma_s^2}} e^{-\frac{\tau^2}{\sigma_t^2}} e^{-\frac{(y_{k,j,t}-y_{k-m,n,t-\tau})^2}{2\sigma_y^2}} y_{k-m,n,t-\tau},
$$

where subscript $t$ denotes the frame number and $\sigma_t$ is the temporal variance. The normalization factors $w_{k,j,t}$ are defined accordingly. Since the pixels are weighted according to their photometric distance from the central pixels, no motion compensation is required (in a sense, the motion estimation is performed explicitly in such an approach).

3.2 Color filtering

For color images in the RGB color space, bilateral filter can be applied straightforwardly on each of the channels in an independent manner. For images in the YUV
representation (which is one of the most commonly used color space in video processing pipelines), there are two possibilities. The simplest way is to apply the filter to the Y (luma) channel only, since it carries the majority of information about the image, and edge artifacts and noise are mostly seen in this channel.

However, often the UV (chroma) channels undergo much greater degradation, due to the tendency of compression algorithm to perform coarser quantization and subsample the chroma in 4:2:0 format. This necessitates the processing of UV channels (see Figure 1). At the same time, the luma channel usually contains the information about the edges in the frame. Therefore, the luma and chroma channels should be coupled in the following way,

\[ \hat{u}_{k,j} = \frac{1}{w_{k,j}} \sum_{m=-P_{uv}}^{P_{uv}} \sum_{n=-P_{uv}}^{P_{uv}} e^{-\frac{m^2+n^2}{2\sigma_y^2}} \left(\frac{(y_{k,j}-y_{k-m,j-n})^2}{2\sigma_y^2}\right) \frac{(u_{k,j}-u_{k-m,j-n})^2}{2\sigma_{uv}^2} u_{k-m,j-n}. \]

Here \( u_{k,j} \) \((k = 1, ..., M, j = 1, ..., N, \) assuming that the chroma channels have the same resolution as the luma) denote the U channel pixels, \( \sigma_y \) and \( \sigma_{uv} \) are the spatial variance parameters of the Y and UV channels, respectively, and \( P_{uv} \) is the window size. \( w_{k,j} \) are normalization factors defined accordingly. The same processing is applied to the V chroma channel. The use of Y-channel weights allows to compensate for the misalignment of luma and chroma edges (which appears as color bleeding) resulting from 4:2:0 subsampling.

Extending the Durand-Dorsey approach to the YUV case, we have

\[ \hat{u}_{k,j}^{c,c'} = \frac{1}{w_{k,j}} \sum_{m=-P_{uv}}^{P_{uv}} \sum_{n=-P_{uv}}^{P_{uv}} e^{-\frac{m^2+n^2}{2\sigma_y^2}} e^{-\frac{(c'-y_{k,j})^2}{2\sigma_y^2}} e^{-\frac{(c'-u_{k,j})^2}{2\sigma_{uv}^2}} u_{k-m,j-n}, \]

where

\[ \hat{g}_{i,j}^{c,c'} = \frac{(c'-y_{i,j})^2}{2\sigma_y^2} e^{-\frac{(c'-u_{i,j})^2}{2\sigma_{uv}^2}} u_{i,j}, \]
Figure 1: Luma and chroma components of the HQV coaster sequence, showing typical artifacts due to strong quantization of the chroma components and 4:2:0 subsampling. Note the coarse resolution of the chroma channel.
Note that now we have two constants, \( c \) and \( c' \). The chroma bilateral filter is approximated as

\[
\hat{u}_{i,j} \approx \sum_{l=0}^{L_{uv}} \sum_{l'=0}^{L'_y} \mu_{i,j} \cdot \hat{u}_{i,j} \Delta_{uv,l} \Delta_{y,l'},
\]

where \( \Delta_{uv} = 255L_{uv}^{-1} \) and \( \Delta_y' = 255L_y'^{-1} \) are the luma and chroma intensity band widths, respectively, and \( \mu_{i,j} = \max \{ 1 - \Delta_{uv}^{-1} |u_{i,j} - l\Delta_{uv}|, 0 \} \cdot \max \{ 1 - \Delta_y'^{-1} |y_{i,j} - l\Delta_y'|, 0 \} \) is a mask determining whether the \((i, j)\)-th pixel belongs to the \( l \)-th luma and \( l' \)-th chroma band.

Since the luma usually has smaller visual significance, a coarser and more aggressive smoothing can be applied to the U and V channels. Particularly, this implies that \( \sigma_{uv} \) can be larger, while \( L_{uv} \) and \( P_{uv} \) smaller. In addition, \( L'_y \) can be smaller than \( L_{uv} \), as the luma channel is used only as an edge indicator. An alternative is to cluster all the possible joint intensity values of Y and U in order to reduce the number of bands.

## 4 Lazy sliding window

The main advantage of SIMD architectures is the ability to compute several results (filtered pixels) at once. In order to take advantage of this parallelism, we introduce the lazy sliding window approach. Instead of looking on one central pixel and a window around it, we have a block of \( 4 \times 4 \) pixels (central block) and slide the window in increments of 4. The window is the same for all the pixels in the central block (Figure 2, bottom). The window size is \( 4(2P_y + 1) \times 4(2P_y + 1) \), where \( P_y \) now denotes the window size in blocks of size \( 4 \times 4 \). Moving the window requires the load of only \( 2P_y + 1 \) blocks of size \( 4 \times 4 \). When \( P_y \) is sufficiently large, the lazy sliding window approach is (asymptotically) identical to the traditional sliding window. The entire \( 4 \times 4 \) blocks of results can be computed simultaneously on a SIMD-type digital signal processor.
Figure 2: Filtering using sliding window (top) and lazy sliding window (bottom). Shown in red and blue are two subsequent positions of the window.
The luma bilateral filter with the Durand-Dorsey acceleration can be approximated in the following way using this approach:

1. Divide the luma image into $4 \times 4$ blocks, $Y_{i,j}$, $i = 1, ..., M/4$, $j = 1, ..., N/4$ (we denote by upper indices the block numbers. For simplicity, $M, N$ are assumed to be divisible by 4).

2. for $l = 0, ..., L_y$ do

3. for every block $Y_{i,j}$ in raster scan order do

4. Compute the non-linearity for the neighbor blocks in a pixel-wise manner,
   \[
   G_{i,j}^{k,j+q} = e^{-\left(\frac{(\Delta_y - Y_{m,n}^{i+k,j+q})^2}{2\sigma_y^2}\right)},
   \]
   with $k, q = -P_y, ..., +P_y$, and $m, n = 1, ..., 4$.

5. Compute the filter output, $h = \sum_{m,n=1}^{P_y} \sum_{k,q=-P_y}^{P_y} G_{i,j}^{k,j+q} \cdot Y_{m,n}^{i+k,j+q}$.

6. Compute the normalization, $w = \sum_{m,n=1}^{P_y} \sum_{k,q=-P_y}^{P_y} G_{i,j}^{k,j+q}$.

7. Compute the mask, $\mu_{i,j}^{m,n} = \max\{1 - \Delta_y^{-1}|Y_{m,n}^{i,j} - l\Delta_y|, 0\}$.

8. Merge $\hat{Y}_{i,j} \leftarrow \hat{Y}_{i,j} + \mu_{i,j}^{m,n} h/w$.

9. end

10. end

Note that Stages 4–8 are mostly SIMD operations with $4 \times 4$ matrices. Another important advantage is that, since the lazy sliding window moves at each step by one block, the computations of Stages 4–7 from previous steps can be reused. If we store the previous values of $G_{i,j}^{k,j}$, we have to perform Stage 4 for the new $2P_y + 1$ blocks only. If in addition we store the sums $\sum_{m,n=1}^{P_y} \sum_{k,q=-P_y}^{P_y} G_{m,n}^{i+k,j+q}$ and $\sum_{m,n=1}^{P_y} \sum_{k,q=-P_y}^{P_y} Y_{m,n}^{i+k,j+q}$, after sliding the window, we have to add only the sums $\sum_{m,n=1}^{P_y} G_{m,n}^{i+k,j+q}$ and $\sum_{m,n=1}^{P_y} Y_{m,n}^{i+k,j+q}$.
The chroma bilateral filter is implemented in a similar way:

\begin{verbatim}
1 Divide the chroma image into $4 \times 4$ blocks, $U^{i,j}$, $i = 1, ..., M/4$, $j = 1, ..., N/4$ (we denote by upper indices the block numbers. For simplicity, $M, N$ are assumed to be divisible by 4).
2 for $l = 0, ..., L_{uv}$ do
3    for $l' = 0, ..., L'_{y}$ do
4        for every block $U^{i,j}$ in raster scan order do
5            Compute the non-linearity for the neighbor blocks in a pixel-wise manner,
6                \[ G_{m,n}^{i+k,j+q} = e^{-\frac{(l'\Delta_{y} - Y_{m,n}^{i+k,j+q})^2}{2\sigma_{y}^2}} e^{-\frac{(l\Delta_{uv} - U_{m,n}^{i+k,j+q})^2}{2\sigma_{uv}^2}}, \]
7                with $k, q = -P_{uv}, ..., +P_{uv}$, and $m, n = 1, ..., 4$.
8            Compute the filter output,
9                \[ h = \sum_{m,n=1}^{4} \sum_{k,q=-P_{uv}}^{P_{uv}} G_{m,n}^{i+k,j+q} \cdot U_{m,n}^{i+k,j+q}. \]
10           Compute the normalization, \[ w = \sum_{m,n=1}^{4} \sum_{k,q=-P_{uv}}^{P_{uv}} G_{m,n}^{i+k,j+q}. \]
11           Compute the mask, \[ \mu_{m,n}^{i,j} = \max\{1 - \Delta_{y}\frac{1}{\sigma_{y}^2}|Y_{m,n}^{i,j} - l'\Delta_{y}'|, 0\} \max\{1 - \Delta_{uv}\frac{1}{\sigma_{uv}^2}|U_{m,n}^{i,j} - l\Delta_{uv}|, 0\}. \]
12           Merge $\hat{U}^{i,j} \leftarrow \hat{U}^{i,j} + \mu_{i,j}^{i,j} h/w$.
13 end
14 end
15 end
\end{verbatim}

5 Results

We tested our implementation of the bilateral filter on several standard sequences. Figure 3 shows a synthetic experiment of noise reduction in a picture contam-
inated by additive Gaussian noise. Figure 4 show noise reduction in video sequence taken from the Silicon Optix Hollywood Quality Video (HQV) benchmarks. Figure 5 shows the advantage of coupled luma-chroma denoising over luma only denoising.

6 Conclusions

We presented an efficient implementation of the bilateral filter on parallel SIMD-type architecture. The main purpose of the bilateral filter is analog noise reduction, removal of MPEG compression artifacts and results of chroma subsampling such as color bleeding. The bilateral filter can be applied on progressive or interlaced video in YUV format.

References


Figure 3: Result of lazy sliding window bilateral filter applied on an image with synthetic noise.
Part of a frame from the HQV flower test sequence (Y channel)

Bilateral filter result

\( \sigma_r = 4, P = 2, L = 20 \)

Figure 4: Result of lazy sliding window bilateral filter on the HQV flower test sequence.
Figure 5: U channel of the color HQV coaster sequence, showing the advantage of coupled luma-chroma denoising. Note the super-resolution and edge sharpening effect.


