Domain Transform for Edge-Aware Image and Video Processing
A faster method of performing edge-preserving filtering on images

Main idea: domain transform from image in $\mathbb{R}^5 (x,y,r,g,b)$ to lower dimension where distances are preserved

Then, perform filtering on the transformed image
How the transformation will look:
2D RGB image $I$: $I : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3$

$I$ defines a 2D manifold $M_I$ in $\mathbb{R}^5$

Let $\hat{p} = (x_p, y_p, r_p, g_p, b_p) \in M_I$

$\hat{p}$ has a corresponding pixel in $I$ with:
- Spatial coordinates $p = (x_p, y_p)$, and
- Range coordinates $I(p) = (r_p, g_p, b_p)$

Let $F(\hat{p}, \hat{q})$ be an edge-preserving filter in 5D

Filtered image $J$ is:

$$J(p) = \int_{\Omega} I(q) F(\hat{p}, \hat{q}) \, dq$$
Example - Bilateral Filter

- J, the image obtained when filtering I with F can be expressed as:
  \[ J(p) = \int_{\Omega} I(q) F(\hat{p}, \hat{q}) \, dq \]

- Bilateral filter kernel is given by:
  \[ F(\hat{p}, \hat{q}) = G_{\sigma_s}(\|p - q\|) \cdot G_{\sigma_r}(\|I(p) - I(q)\|) \]

  \( p = (x_p, y_p) \) and \( I(p) = (r_p, g_p, b_p) \)
Does there exist:

• Transformation $t : \mathbb{R}^5 \rightarrow \mathbb{R}^l$, $l < 5$, and

• Filter kernel $H$ defined over $\mathbb{R}^l$, such that

• For any input image $I$, an equivalent result to the 5D edge-preserving kernel $F$ is produced, i.e.:

$$J(p) = \int_{\Omega} I(q) F(\hat{p}, \hat{q}) dq = \int_{\Omega} I(q) H(t(\hat{p}), t(\hat{q})) dq$$
Instead of starting with $t : \mathbb{R}^5 \to \mathbb{R}^l, l < 5$, let's try to find $t : \mathbb{R}^2 \to \mathbb{R}$, where

- $t$ preserves (in $\mathbb{R}$) the original distances between points $(x_i, I(x_i))$ given by some distance metric (e.g. Euclidean):

$$|t(x_i, I(x_i)) - t(x_j, I(x_j))| = \|(x_i, I(x_i)) - (x_j, I(x_j))\|$$
Let $ct(x) = t(\hat{x}) = t(x, I(x))$

Transform must satisfy

$$ct(x + h) - ct(x) = h + |I(x + h) - I(x)|$$

in order to preserve L1 distances between neighboring pixels $x$ and $x + h$ (with sampling width $h$)
Dividing by \( h \) and taking the limit as \( h \) approaches 0 gives:

\[
ct'(x) = 1 + |I'(x)|
\]

Integrating both sides gives:

\[
ct(u) = \int_{0}^{u} 1 + |I'(x)| \, dx, \quad u \in \Omega
\]
Finding t (continued)

Then, distance between two points in new domain is given by:

\[ ct(u) = \int_0^u 1 + |I'(x)| \, dx, \quad u \in \Omega \]

\[ ct(w) - ct(u) = \int_u^w 1 + |I'(x)| \, dx \]

the L1 arc length of curve C in the interval [u,w]
Generalizing t - Multichannel

- Want to do this with RGB images, not just grayscale; so we want $t : \mathbb{R}^4 \rightarrow \mathbb{R}$
- Change from this:

\[
ct(u) = \int_0^u 1 + |I'(x)| \, dx, \quad u \in \Omega
\]

- To this:

\[
ct(u) = \int_0^u 1 + \sum_{k=1}^{c} |I'_k(x)| \, dx
\]

(sum over all $c = 3$ color channels)
Generalizing \( t \) – Multidimensional

- Want to do this with 2D images, not just 1D signals
- Unfortunately, not possible in general
- So, instead we apply our current 1-dimensional \( t \) to rows/columns of image
  - First, along each row
  - Then, along each column
  - Iterate \( N \) times
  - Good \( N \) depends on the geometry of the image
More iterations preserves edges somewhat better:

(a) Input  (b) 1 itr.  (c) 3 itr.
Bilateral kernel has spatial vs. range parameters $\sigma_s$ and $\sigma_r$:

$$F(\hat{p}, \hat{q}) = G_{\sigma_s}(||p - q||) G_{\sigma_r}(||I(p) - I(q)||)$$

We can encode these into $ct$ by adding factor $\sigma_s / \sigma_r$:

$$ct(u) = \int_0^u 1 + \frac{\sigma_s}{\sigma_r} \sum_{k=1}^c |I_k'(x)| \, dx$$
Wanted $t$ and $H$ such that:

\[
J(p) = \int_{\Omega} I(q) F(\hat{p}, \hat{q}) \, dq = \int_{\Omega} I(q) H(t(\hat{p}), t(\hat{q})) \, dq
\]

- Found $t$
- $H$ can be any filter whose response decreases with distance at least as fast as $F$’s
- Choices of $H$: Normalized Convolution, Interpolated Convolution, Recursive Filtering
Normalized Convolution

- One of three filters the paper describes

\[ J(p) = \frac{1}{K_p} \sum_{q \in D(\Omega)} I(q) H(t(\hat{p}), t(\hat{q})) \]

\[ H(t(\hat{p}), t(\hat{q})) = \delta\{ |t(\hat{p}) - t(\hat{q})| \leq r \} \]

\[ r = \sigma_H \sqrt{3} \]

- Parallelizable – fast GPU implementation
Results - Smoothing Quality

(a) Input  (b) BF, $\sigma_s = 17$  (c) NC, $\sigma_s = 17$  (d) BF, $\sigma_s = 40$  (e) NC, $\sigma_s = 80$
Filtering on CPU - NC w/ 3 iterations
- 1 megapixel: 0.16 seconds
- 10 megapixels: 1.6 seconds
- 3.3x speedup with quad-core CPU
- Vs. CTBF: 10 seconds with 1/3 the work (single color channel instead of all 3)

Filtering on GPU
- 1 megapixel: 0.007 seconds
- Speedup of 23x vs. single core CPU implementation
- Vs. WLS: 1 second for grayscale image
Figure 12: Fine detail manipulation. (a) Input image. (b) Our result. $J_1$ was obtained with the IC filter ($\sigma_s = 20$ and $\sigma_r = 0.08$). (c) EAW result by Fattal [2009].
Input:
http://www.youtube.com/watch?v=HsAW9sh_IW0&hd=1

Output (1080p video filtered in real time):
http://www.youtube.com/watch?v=lTy9W5mWG_0&hd=1