Fast Bilateral Solver

Projet du Cours Introduction à l'Imagerie Numérique

Tong Zhao, Clement Riu

Jan 2019

Ecole des Ponts et Chaussées

Outline

1. Motivation
2. Bilateral Grid
3. Bilateral Solver [1]
4. Experiments
5. Conclusion

Motivation

Bilateral Filter

Filtering : apply a local function to the image using a sum of weighted neighboring pixels.

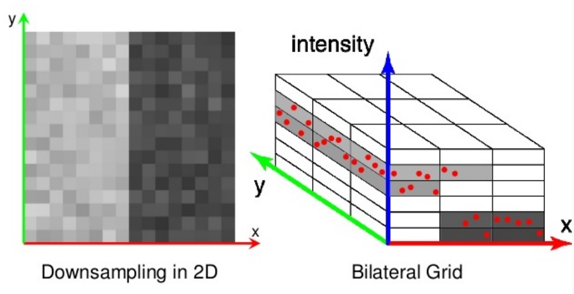
Bilateral Filter :

$$I^{filtered}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

Bilateral Grid

Bilateral Grid [2]

Bilateral Space : a high dimensional space containing both the spatial and the color information of pixels.



A grayscale image → 3D space
An RGB image → An YUV image → 5D space

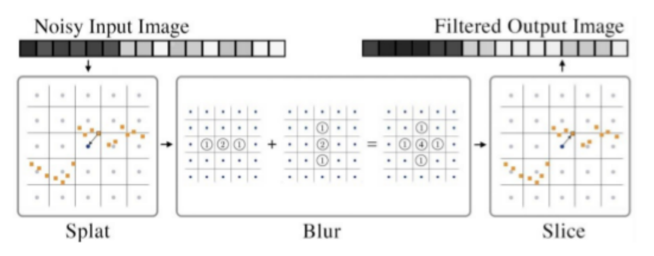
Basic Operations

Splat : Project the image to the high dimensional bilateral space.

Blur : Perform a convolution on bilateral space.

Slice : Retreive the values of pixels by interpolation.

$$W = S^T \overline{B} S$$



Bilateral Solver [1]

Formulation

Given a reference image x, a target image t and a confidence map c, we minimize the following problem :

$$\min_x \frac{\lambda}{2} \sum_{i,j} \hat{W}_{i,j}(x_i - x_j)^2 + \sum_i c_i(x_i - t_i)^2 \tag{1}$$

where

$$W_{i,j} = \exp\left(-(d_{spatial})^2_{i,j} - (d_{luma})^2_{i,j} - (d_{chroma})^2_{i,j}\right) \tag{2}$$

Tricks

Variable Substitution A voxel in bilateral space is mapped to multiply tuples by using $(\sigma_{spatial}, \sigma_{luma}, \sigma_{chroma})$ as the discretization step.

$$\mathbf{x} = S^T \mathbf{y} \tag{3}$$

$$W_{i,j} = \exp\left(-\frac{(d_{spatial})^2_{i,j}}{2\sigma^2_{spatial}} - \frac{(d_{luma})^2_{i,j}}{2\sigma^2_{luma}} - \frac{(d_{chroma})^2_{i,j}}{2\sigma^2_{chroma}} \right) \tag{4}$$

Bistochastization Repeat row and column normalization until convergence, so that all rows and columns have the same mean value.

$$\hat{W} = S^T D_{\mathbf{m}}^{-1} D_{\mathbf{n}} \overline{B} D_{\mathbf{n}} D_{\mathbf{m}}^{-} 1 S \quad \text{with } SS^T = D_{\mathbf{m}} \tag{5}$$

Sparse Linear System

Now we can reformulate the bilateral solver loss function in a quadratic form :

$$\min_{y} \frac{\lambda}{2} y^T A y - b^T y + c \tag{6}$$

where

$$A = \lambda(D_m - D_n \overline{B} D_n) + diag(Sc)$$
$$b = S(c \circ t)$$
$$c = \frac{1}{2}(c \circ t)^T t$$

The minimization is equivalent to solving a sparse linear system : $Ay = b$.

Solver

Preconditioning A simple Jacobi preconditioner is used to accelerate the convergence of the algorithm and to prevent numerical issues.

$$M^{-1} = \mathrm{diag}(A)^{-1}$$

Initialization

$$y_{init} = S(c \circ t)/S(c)$$

Conjugate Gradient Method An iterative optimization algorithm. It is implemented in scipy.

Experiments

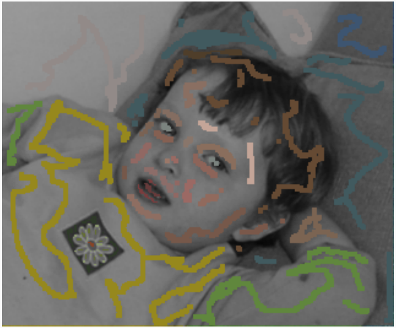
Task : Colorize a grayscale image given a small user-colorized set.

Algorithm 2 Colorisation de l'image L

Input : Image de référence L, image cible L_t, confiance L_c
Output : Image colorée L_o

procedure COLORISATION
$L_t^{yuv} \leftarrow \text{rgb2yuv}(L_t)$
$L^{yuv} \leftarrow \text{rgb2yuv}(L)$
$L_t^u \leftarrow L_t^u - 128$
$L_t^v \leftarrow L_t^v - 128$
$U \leftarrow \text{bilateral-solver}(L,\, L_t^u,\, L_c)$
$V \leftarrow \text{bilateral-solver}(L,\, L_t^v,\, L_c)$
$L_o \leftarrow \text{yuv2rgb}(L^y,\, U + 128,\, V + 128)$
$L_o \leftarrow \dfrac{L_o - L_o^{min}}{L_o^{max}} \times 255$
end procedure

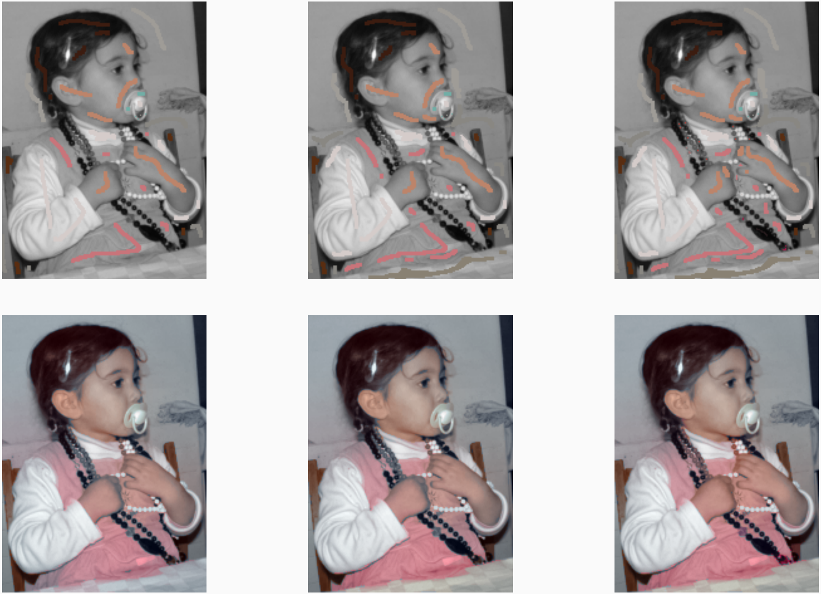
Example : Boy



Example : Monaco

Example : hair

Parameter Tuning : Number and position of colored pixels



Parameter Tuning : $\sigma_{spatial}$

$\sigma_{spatial} = \{1, 2, 4, 8, 16\}$, *respectively*.

Parameter Tuning : σ_{luma}

$$\sigma_{luma} = \{1, 2, 4, 8, 32\}, \textit{respectively}.$$

Parameter Tuning : σ_{chroma}

$$\sigma_{chroma} = \{1, 4, 8, 16, 32\}, \textit{respectively}.$$

$\lambda = \{1, 10, 50, 128, 500\}$, *respectively*.

Parameter Strategy

- σ_{chroma} should be large in order to accelerate the calculation.
- σ_{luma}, σ_{chroma} and λ should be chosen carefully
- The colorized pixels are better to be placed around edges and complicated zones.

Calculation Time

We compared our implementation with the one provided by the author. Although our version is cleaner and more readable, the author's version is very efficient.

To solve a stereo problem, our implementation takes 13s, while the official version takes only 1.69s.

The most time-consuming part in our implementation is the construction of the voxel ($\sim$12s).

`https://github.com/Tong-ZHAO/Fast-Bilateral-Filter`

Conclusion

Conclusion

- Bilateral solver is an edge-aware smoothing algorithm that combines the flexibility and computational-efficiency.
- One needs to choose carefully the parameters to get a good result.

References I

J. T. Barron and B. Poole.
The fast bilateral solver.
In *European Conference on Computer Vision*, pages 617–632. Springer, 2016.

J. Chen, S. Paris, and F. Durand.
Real-time edge-aware image processing with the bilateral grid.
In *ACM Transactions on Graphics (TOG)*, volume 26, page 103. ACM, 2007.