Fully Connected Guided Image Filtering

Longquan Dai Mengke Yuan Feihu Zhang Xiaopeng Zhang
NLPR-LIAMA, Institute of Automation Chinese Academy of Sciences, BEIJING, CHINA
{longquan.dai, yuanmengke2015}@ia.ac.cn, hi.yexu@gmail.com, xiaopeng.zhang@ia.ac.cn

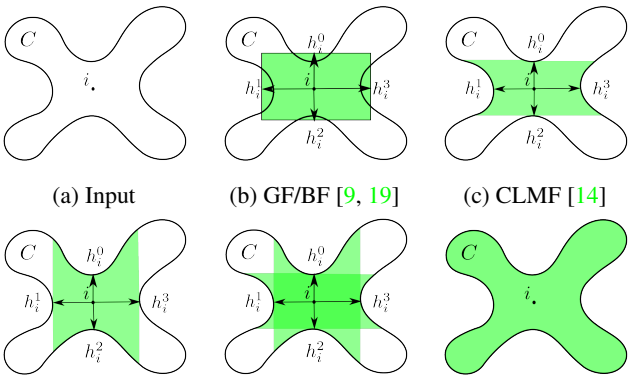
Abstract

This paper presents a linear time fully connected guided filter by introducing the minimum spanning tree (MST) to the guided filter (GF). Since the intensity based filtering kernel of GF is apt to overly smooth edges and the fixed-shape local box support region adopted by GF is not geometric-adaptive, our filter introduces an extra spatial term, the tree similarity, to the filtering kernel of GF and substitutes the box window with the implicit support region by establishing all-pairs-connections among pixels in the image and assigning the spatial-intensity-aware similarity to these connections. The adaptive implicit support region composed by the pixels with large kernel weights in the entire image domain has a big advantage over the predefined local box window in presenting the structure of an image for the reason that: 1, MST can efficiently present the structure of an image; 2, the kernel weight of our filter considers the tree distance defined on the MST. Due to these reasons, our filter achieves better edge-preserving results. We demonstrate the strength of the proposed filter in several applications. Experimental results show that our method produces better results than state-of-the-art methods.

1. Introduction

Edge-preserving image filters are fundamental building blocks for many computer vision and graphics applications [4, 10, 13] because they smooth out trivial details while preserving major image structures (edge-preserving). A common approach is to pose this problem as the weighted average $p_i = \sum_{j\in\Omega_i} W_{ij}q_j$ over an explicit local support region Ω_i around the pixel i, where W_{ij} denotes the weight kernel, q_i and p_i are the intensities of the input image q and the output image I at i, respectively.

Traditional edge-aware filtering methods often explicitly limit the support region Ω_i to a box window and compute W_{ij} from both intensity similarity and spatial affinity or only from the intensity similarity between i and $j \in \Omega_i$. Since the shape of the most relevant pixels in Fig.1a is irregular, the box window in Fig.1b cannot efficiently capture the most relevant pixels of i in the image domain. To solve this problem, people designed various geometric-adaptive support regions. However, these sophisticated algorithms still cannot pick out all the most relevant pixels as shown in Figs.1c 1d 1e. The shortcoming limits the ability of these algorithms to model long-range connections as well as the complicated geometric structures of an image and therefore results in unsatisfactory smoothing results. But the explicit support region cannot be easily removed because 1, some filters do not consider the spatial affinity, the local support region is used to to block the interconnection of distant pixels; 2, some filters' computational complexity depends on the size of support regions, a small window is chosen to reduce the computational cost.



(a) Input (b) GF/BF [9, 19] (c) CLMF [14]

(d) CLMF [14] (e) MLPA [18] (f) Ours

Figure 1: Support regions of a pixel i in the region component C. (a) is input image. (b) (c) (d) (e) are the explicit support regions determined by BF, GF, CLMF and MLPA, where (c) (d) show the support regions produced by different scanning orders. (f) is the implicit support region of ours.

In order to exploit all information in arbitrary shapes, we can compute all-pairs-weights W_{ij} of a filter for pixels i, j in the entire image domain Ω and consider that the implicit support region $\tilde{\Omega}_i = \{j \in \Omega \mid W_{ij} > T\}$ is versatile and can present the ideal support region as shown in Fig.1f, where T is a threshold and a plausible value of T is 0.001. Since $\tilde{\Omega}_i$ contains all pixels with large weights, we have $p_i = \sum_{j\in\tilde{\Omega}_i} W_{ij}q_j \approx \sum_{j\in\Omega} W_{ij}q_j$ which implies that the fully connected filtering $\sum_{j\in\Omega} W_{ij}q_j$ is a good approxima-

tion of the ideal filtering $\sum_{j\in\tilde{\Omega}_i} W_{ij}q_j$. From this implicit support region filtering perspective, the problem of the explicit support region is that it limits the possible points that compose the implicit support region in a local area. This drawback will produce negative results in some situations. To solve this, we ought to exploit the fully connected filtering scheme to compute the ideal filtering result p_i.

The idea of the fully connected filtering is not new. Incorporating optimization, Xu took it for texture removing [21] and image editing [20]. But his optimization based methods are very slow. Yang applied it to stereo matching [22]. But his method only considers the intensity similarity, which leads to that two distant pixels with similar intensities always have a large similarity. The bilateral filter (BF) [19] cooperates with distance-intensity-aware weights in a box support region Ω_i^b to smooth an image. We can simply relax Ω_i^b to the entire image domain Ω to perform the fully connected bilateral filtering (FCBF). Based on Yang'method and FCBF, Bao *et al*. [1] define the tree filter (TF) for texture smoothing. However, the computational burden of BF is proportional to the size of Ω_i^b and limits the applications of FCBF and TF in practical environment. The guided filter (GF) [9] is considered to be the first edge-preserving filter whose computational complexity does not depend on the size of the box support region Ω_i^b. But, it also disregards the spatial affinity at all. When we extend Ω_i^b to Ω and perform the fully connected guided filtering, sharp edges are more likely to be blurred (*i.e*. the implicit support region defined by the filtering kernel of GF is problematic).

Our main contribution in this paper is to design a new GF-like filter that considers both spatial affinity and intensity similarity and performs the fully connected filtering in linear time. Additionally, the fully connected filtering can aid original GF to choose the most relevant pixels according to the underlying structure of an image. These improvements can greatly expand GF's application scope.

2. Related work

GF disregards the spatial affinity at all. We can observe this from the kernel (1) of GF clearly.

$$W_{ij}^{gf} = \frac{\sum_{k\in\Omega_i^b\cap\Omega_j^b}\left(1+\frac{(I_i-E_{\Omega_k^b}(I))(I_j-E_{\Omega_k^b}(I))}{D_{\Omega_k^b}(I)+\epsilon}\right)}{(\sum_{k\in\Omega_i^b}1)(\sum_{k\in\Omega_j^b}1)} \quad (1)$$

where $E_{\Omega_i^b}(x)$ and $D_{\Omega_i^b}(x)$ denote the arithmetical average and variance of x in the explicit local support region Ω_i^b. The drawback leads GF to smooth the sharp boundaries in applications such as depth upsampling [15] and stereo matching [10]. To avoid over-flatten artifacts, GF has to employ a small support region to block the interconnections between distant pixels. The filtering results of GF is computed from a two steps procedure in the local multipoint filtering framework [11]. The first multipoint estimation step assumes $\Re(\boldsymbol{\alpha}_i, I_j) = \alpha_i^1 I_j + \alpha_i^2$ for $j \in \Omega_i^b$ and exploits objective function (2) to determine coefficients $\boldsymbol{\alpha}_i = (\alpha_i^1, \alpha_i^2)$.

$$E(\boldsymbol{\alpha}_i) = \sum\nolimits_{j\in\Omega_i^b}(\Re(\boldsymbol{\alpha}_i, I_j) - q_j)^2 + \epsilon(\alpha_i^1)^2 \quad (2)$$

where q represents the input image. The closed form solution of the quadratic objective function is given by

$$\alpha_i^1 = \frac{E_{\Omega_i^b}(Iq) - E_{\Omega_i^b}(I)E_{\Omega_i^b}(q)}{D_{\Omega_i^b}(I)+\epsilon} \quad (3)$$

$$\alpha_i^2 = E_{\Omega_i^b}(q) - \alpha_i^1 E_{\Omega_i^b}(I) \quad (4)$$

As each given pixel i has a number of multipoint estimates $\{\Re(\boldsymbol{\alpha}_i, I_j) \mid j \in \Omega_i^b\}$ for each fixed $\boldsymbol{\alpha}_i$, the final filtering result p is defined as the average of these multipoint estimates in the second aggregation step:

$$p_i = \frac{\sum_{j\in\Omega_i^b}\Re(\boldsymbol{\alpha}_i, I_j)}{\sum_{j\in\Omega_i^b}1} = E_{\Omega_i^b}(\alpha^1)I_i + E_{\Omega_i^b}(\alpha^2) \quad (5)$$

Eqs.(2) (5) equally treat all pixels in the local support region and Eqs.(3) (4) perform arithmetical average. This is another irrational way of GF as the contribution of each pixel to the center pixel should correspond with the predefined similarity. So one reasonable way is to introduce affinity weights to these formulae. However, people prefer the approach that transforms explicit box support regions to more sophisticated explicit support regions.

In literature, Lu [14] first points out that the box support region Ω_i^b is not geometric-adaptive. To obtain structure-aware support regions, he adaptively constructs a cross based support region Ω_i^c with four varying arm lengths $\{h^1, h^2, h^3, h^4\}$ as illustrated in Figs.1c 1d. The smoothing procedure of his cross-based local multipoint filter (CLMF) is similar to GF. In the first step, CLMF replaces Ω_i^b by Ω_i^c. In the second step, CLMF computes the result p_i by

$$p_i = \frac{\sum_{j\in\Omega_i^c}\sum_{k\in\Omega_j^c}\Re(\boldsymbol{\alpha}_i, I_j)}{\sum_{j\in\Omega_i^c}\sum_{k\in\Omega_j^c}1} \quad (6)$$

Tan [18] argues that CLMF is problematic yet. The support region of CLMF depends on the scanning order as illustrated in Figs.1c 1d and the complexity of calculation for support arms at each point depends on the maximum length of arms. These lead to unreliable estimation especially for points near the concave region and slow down the algorithm. Even worse, both CLMF and GF only consider the intensity information in the guidance image regardless of the position of the point in the image, which is likely to overly smooth the input image in regions where the values of the guidance image are similar. To solve these problems, Tan designs a local polynomial approximation based multipoint filter (MLPA). MLPA employs a

2D quadratic spatial regularization term $\mathfrak{S}(\boldsymbol{\beta}_i, x_j, y_j) = \beta_i^0 + \beta_i^1 x_j + \beta_i^2 y_j + \beta_i^3 x_j y_j + \beta_i^4 x_j^2 + \beta_i^5 y_j^2$ to grant it the distance-aware ability, where x_j and y_j are the coordinates of the pixel j and $\boldsymbol{\beta}_i = (\beta_i^0, \beta_i^1, \beta_i^2, \beta_i^3, \beta_i^4, \beta_i^5)$. In the first multipoint estimation step, MLPA employs optimization (7) which jointly exploits $\mathfrak{R}(\boldsymbol{\alpha}_i, I_j)$ and $\mathfrak{S}(\boldsymbol{\beta}_i, x_j, y_j)$ to find coefficients $\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i$.

$$E(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i) = \sum\nolimits_{j \in \Omega_i^m} w_{ij} (\mathfrak{R}(\boldsymbol{\alpha}_i, I_j) + \mathfrak{S}(\boldsymbol{\beta}_i, x_j, y_j) - q_j)^2 + \epsilon_r (\alpha_i^1)^2 + \epsilon_s \sum\nolimits_{m=1}^{5} (\beta_i^m)^2 \tag{7}$$

Here $\Omega_i^m = \Omega_i^H \bigcup \Omega_i^V$, Ω_i^H denotes the support region of the pixel i with the horizontal direction first aggregation. Similarly, Ω_i^V is the support region of the pixel i with the vertical direction first aggregation, and w_{ij} is defined as

$$w_{ij} = \begin{cases} 1 & j \in (\Omega_i^H - \Omega_i^V) \bigcup (\Omega_i^V - \Omega_i^H) \\ 2 & j \in \Omega_i^H \bigcap \Omega_i^V \end{cases} \tag{8}$$

Similar to Eq. (6), MLPA determines the result p_i by

$$p_i = \frac{\sum_{j \in \Omega_i^m} \sum_{k \in \Omega_j^m} w_{ij} (\mathfrak{R}(\boldsymbol{\alpha}_i, I_j) + \mathfrak{S}(\boldsymbol{\beta}_i, x_j, y_j))}{\sum_{j \in \Omega_i^m} \sum_{k \in \Omega_j^m} w_{ij}} \tag{9}$$

3. Motivation & Technique background

MLPA is also improvable. First, the extra spatial regularization term $\mathfrak{S}(\boldsymbol{\beta}_i, x_j, y_j)$ increases MLPA's computational cost. Specifically, MLPA inverses a 7×7 positive matrix as the quadric objective function (7) totally contains 7 independent variables. In contrast, GF and CLMF inverse a 2×2 matrix. The distance-aware ability of MLPA is at the cost of increasing the size of the positive matrix nearly four times. Second, although the support region of MLPA is obtained in data-driven manner, it still cannot contain all the most relevant pixels in a complicated concave support region as shown in Fig.1e and therefore loses much valuable information. Third, it is a pity that MLPA also treats all pixels in Eq. (7) equally. (The choice $w_{ij} = 2$ is only used to define the aggregation order invariant support region Ω_i^c without the function of indicating importance.)

These problems can be fundamentally solved under the fully connected filtering framework. We establish a full connection between each pixel and the remaining pixels in Ω. For each pixel i, we assign a weight to a pixel $j \in \Omega$ to denote the affinity between i and j. Employing the weight, objective functions (2) (7) are able to consider all information of an image. The approach is significantly different from GF, CLMF and MLPA that only exploit the information in the explicit local support region and equally treats every pixel in Eqs.(2) (7) and the arithmetical average operator $E_{\Omega_i}(x)$. Moreover, if the weight is distance-aware, the newly defined filter will automatically inherit the spatial-affinity and no longer needs to add an extra spatial regularization term $\mathfrak{S}(\boldsymbol{\beta}_i, x_j, y_j)$ which increases the computational cost. Additionally, the new filter replaces the explicit support region by the implicit support region and therefore solves the second and third problems in above section. However, finding appropriate weights for edges of full connections is not an easy task as the weight w_{ij} should satisfy:

1. w_{ij} measures the distance of two pixels on the intensity surface of an image. For two pixels on either side of an edge, it assigns a small weight to them. For two pixels on the same side of an edge, it gives a large weight. More importantly, for two far pixels, the weight approaches to zero no matter whether they are on the same side of an edge.

2. There is a fast algorithm to compute the fully connected aggregation $C_i = \sum_j w_{ij} x_j$ because solving objective functions (2) (7) and calculating the filtering result p_i (5) (6) (9) depend on it.

Recently, many geodesic-like affinities [8, 12, 24] have been proposed, but only the tree affinity [22] satisfies the two constraints. The affinity is defined on a minimum spanning tree (MST). By treating an image as an undirected 4-connected grid with edges measured by the sum of color differences between adjacent pixels, an MST can be computed by removing edges with large intensity difference and leaving the remaining edges connecting through all pixels as a tree. MST is geometric-aware because the sum of total difference is the minimum of all spanning trees and edges with large intensity difference will be removed during spanning tree construction. Hence it is reasonable to define a spatial affinity along the MST.

The tree distance d_{ij}^t in an MST is the sum of lengths of the edges between pixels i and j, where the length of an edge is the Euclidean distance between the two connected pixels. The spatial similarity w_{ij}^t of i and j is given by

$$w_{ij}^t = \exp(-d_{ij}^t / \sigma) \tag{10}$$

where σ is a constant used to adjust the similarity. Yang [22] proves that the fully connected aggregation $\sum_j w_{ij}^t x_j$ can be efficiently computed by traversing the tree structure of the MST in two sequential passes. Initially, we visit the nodes of the MST using the breadth first traversal algorithm to assign each node v_n an order n based on the visit sequence with the property that if v_m is the parent of v_n, then $m \leqslant n$, where $1 \leqslant m, n \leqslant M$ and M denotes the nodes number of the MST. In the first pass, while the tree is traced from v_M to v_1, ${C^{\uparrow}}_{v_M}$ is not updated until all its children have been updated by the updating formula (11).

$${C^{\uparrow}}_{P(v_n)} = C_{P(v_n)} + w_{v_n, P(v_n)}^t {C^{\uparrow}}_{v_n} \tag{11}$$

Time		GF [9]	CLMF-0 [14]	CLMF-1 [14]	MLPA-0 [18]	MLPA-1 [18]	MLPA-2 [18]	FCGF-0	FCGF-1	FCBF	TF [1]
	r=5	850 ms	630 ms	1680 ms	2650 ms	3450 ms	6140 ms	**930 ms**	**1730 ms**	40 s	40 s
	r=10	850 ms	1320 ms	2130 ms	2650 ms	3450 ms	6140 ms	**930 ms**	**1730 ms**	40 s	40 s

Table 1: Running time comparison of different methods with different window sizes.

where $C_{v_m} = x_{v_m}$, $P(v)$ records the father of v and ${C^{\uparrow}}_{v_m}$ is the intermediate aggregation result. In the second pass, we employ the updating formula (12) to renew $C(v)$ when the tree is traversed from v_1 to v_M.

$$C_{v_n} = w^t_{P(v_n),v_n} {C^{\uparrow}}_{P(v_n)} + (1 - (w^t_{v_n,P(v_n)})^2) {C^{\uparrow}}_{v_n} \quad (12)$$

The updating algorithm is very useful and can be applied in many computer vision applications [1, 3, 22]. In this paper, we employ Eqs.(11) (12) to perform our fully connected guided image filtering.

4. Fully Connected Guided Filtering

In order to arm GF with the fully connected filtering, we design the fully connected guided filter (FCGF) by relaxing the explicit local support region to the entire image domain Ω and introducing w^t_{ij} to GF. Following the local multipoint filtering framework [11], we take the fully connected regression model (13) to compute linear coefficients $\boldsymbol{\alpha}_i = (\alpha^1_i, \alpha^2_i)$ in the first multipoint estimation step. Note that the regression is different from the regression models (2) (7) in which the explicit local support region separates pixels into two categories and assigns the same weight to the pixels in the same category.

$$E(\boldsymbol{\alpha}_i) = \sum\nolimits_{j\in\Omega} w^t_{ij} (\mathfrak{R}(\boldsymbol{\alpha}_i, I_j) - q_j)^2 + \epsilon(\alpha^1_i)^2 \quad (13)$$

Let $E^w(x)$ (16) and $D^w(x)$ (17) denote the weighted average and the weighted variance of x in the entire image domain Ω, the closed form solution can be written as

$$\alpha^1_i = \frac{E^w(Iq) - E^w(I)E^w(q)}{D^w(I) + \epsilon} \quad (14)$$

$$\alpha^2_i = E^w(q) - \alpha^1_i E^w(I) \quad (15)$$

$$E^w(x) = \frac{\sum_{j\in\Omega} w^t_{ij} x_j}{\sum_{j\in\Omega} w^t_{ij}} \quad (16)$$

$$D^w(x) = E^w(x^2) - (E^w(x))^2 \quad (17)$$

As the index j in the linear model $\mathfrak{R}(\boldsymbol{\alpha}_i, I_j)$ is chosen from the entire image domain Ω, we have multiple estimates for point i from regularization results around different j. Hence, in the aggregation step, the filtering result p_i of pixel i is given as a weighted average over these estimates:

$$p_i = \frac{\sum_{j\in\Omega} w^t_{ij} \mathfrak{R}(\boldsymbol{\alpha}_i, I_j)}{\sum_{j\in\Omega} w^t_{ij}} = E^w(\alpha^1) I_i + E^w(\alpha^2) \quad (18)$$

The fully connection and its weights w^t_{ij} distinguish Eqs.(14) (15) (18) from their counterparts Eqs.(3) (4) (5) (6) (9). Note that Eqs.(14) (15) (18) depend on $E^w(x)$ computed by three steps: 1) for each pixel we establish a full connection with the remaining pixels; 2) we assign weights to linking edges according to the tree distance between each given pixel and its connected pixels; 3) we aggregate the contribution of every pixel based on the weights. Although $E^w(x)$ is similar to the local arithmetical average $E_{\Omega_i}(x)$ used by other GF-like filters, the meaning is different. First, $E^w(x)$ takes all of the information in the entire image domain because the explicit local support region is removed. Second, the arithmetical aggregation is transformed to the weighted aggregation by w^t_{ij}. Last but not the least, the pixels with large weights compose the implicit support region and the implicit support region is more sensitive to the underlying geometric structure than the explicit geometric-adaptive support regions shown in Fig.1.

Besides the linear model $\mathfrak{R}(\boldsymbol{\alpha}_i, I_j)$, CLMF and MLPA take a constant model $\mathfrak{C}(\alpha^1_i) = \alpha^1_i$. Putting the model into Eq.(13), we can define the FCGF-0 filter. Similarly, Eqs.(14) (15) (18) taking a linear model are called as FCGF-1. In addition, FCGF can be easily extended to color images. When the filtering input q is multichannel, we can apply the filter to each channel independently. When the guidance image I is multichannel, we follow the way of GF and rewrite the local linear model $\mathfrak{R}(\boldsymbol{\alpha}_i, I_j) = \alpha^1_i I^1_j + \alpha^2_i I^2_j + \alpha^3_i I^3_j + \alpha^4_i$. The filter formulae of FCGF-0 and the color version FCGF are similar to the corresponding versions of CLMF and MLPA. Interested readers can refer to [14, 18] for more details.

Our filter also has a connection with Yang's aggregation method [22] and the tree filter (TF) [1] as all the three methods are built on an MST. Yang first proposes Eqs.(11) (12) and employs them to perform aggregation in stereo matching. However, his $w^t_{v_n,P(v_n)}$ in Eqs.(11) (12) is computed from the intensity difference instead of the Euclidean distance. Moreover, the aggregation operation is not weighted average and thus cannot be directly applied to smooth image. Bao *et al.* [1] exploit Eqs.(11) (12) to define the tree-mean filter (TMF) which is equivalent to our weighted average operator $E^w(x)$ and sequently uses TMF and FCBF to produce the TF filtering result. However, TF costs more running time than our filter because of FCBF.

5. In-depth analysis

In this section, we discuss the distance-aware weights of FCGF as well as its implicit geometric-adaptive support region and show that these advantages come very cheap as the complexity of FCGF is linear.

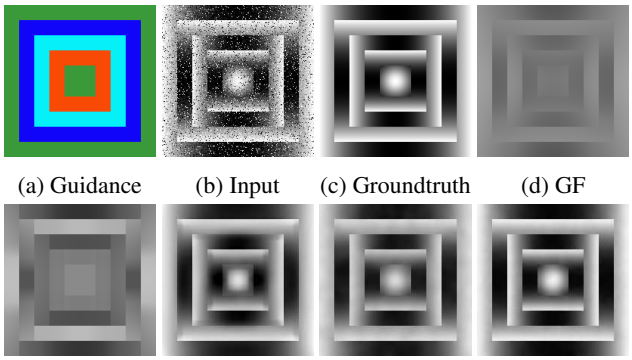
(a) Input (b) GF (c) CLMF-1 (d) MLPA-2 (e) FCBF (f) TF (g) FCGF-1

Figure 2: Filtering kernels computed by GF, CLMF-1, MLPA-2, FCBF, TF and FCGF-1. The amplitudes of GF and CLMF-1 are nearly same. On the contrary, MLPA-2, FCBF, TF and FCGF-1 attenuate amplitudes with increasing distance.

5.1. A linear time filtering algorithm

The computational cost of our algorithm consists of two parts: one is the cost of finding an MST, the other is the cost for data aggregation. The computational complexity of Eqs.(14) (15) (18) depends on the complexity of the weighted average operator $E^w(x)$. Further, $E^w(x)$ needs to compute the fully connected aggregation $C_i = \sum_j w_{ij}^t x_j$. Therefore, the overall complexity is determined by the complexity of $C_i = \sum_j w_{ij}^t x_j$. As detailed in [22], the whole cost aggregation process can be completed by two pass travel along MST using Eqs.(11) (12). The aggregation algorithm is in extremely low computational complexity: only a total of 2 addition/subtraction operations and 3 multiplication operations are required for each pixel. Therefore, FCGF can be computed in linear time. As for finding an MST, Bao [1] proposed a linear time algorithm. Even we choose the nonlinear Kruskal algorithm [2], the algorithm still does not degrade the speed of our filter because the running time is dominated by inverting a symmetric matrix. For instance, FCGF-1 only needs to invert a 2×2 matrix which is a low cost operation and the running time approaches to GF.

We compare our FCGF with GF, CLMF, MLPA, FCBF and TF on a laptop with a 2.0 GHz CPU and 4GB RAM, where FCGF and CLMF contain two filters, MLPA is consisted of three filters, and all filters are implemented with C++ without SIMD instructions. Table 1 lists the running time for filtering a 1-megapixel full-color image. The running times of MLPA and FCGF are higher than GF as they adopt the geometric-adaptive explicit/implicit support region. Although the computational cost of the special case CLMF-0 of CLMF is the lowest in the small window test, the running time of its support region determining method depends on the size of support region. We can expect the computational cost of CLMF becomes larger with increasing support region size. In contrast, MLPA's adaptive support region determining method is not affected by the size of the support region, but its filtering speed is heavily decelerated by the regularization term $\mathfrak{S}(\boldsymbol{\beta}_i, x_j, y_j)$. As for FCGF, its overall running cost is almost same as GF and does not depend on the adaptive support region size. In contrast, the speed of the brute-force implementations of TF and FCBF is extremely slow as TF bases on FCBF and FCBF relaxes the box support region Ω_i^b of BF to Ω.



(a) Guidance (b) Input (c) Groundtruth (d) GF

(e) CLMF-1 (f) MLPA-2 (g) TF (h) FCGF-1

Figure 3: Fully connected filtering. (a) is the guidance image I. (b) is the corrupted input q. (c) is the groundtruth. (d) is the result of GF ($\epsilon = 0.1^2$). (e) is the result of CLMF-1 ($\tau = 25, \epsilon = 0.1^3$). (f) is the result of MLPA-2 ($k = 0.001/255, \epsilon_s = 0.01^2, \epsilon_r = 0.05^2$). (g) is the result of TF ($\sigma = 0.01, \sigma_r = 0.05, \sigma_s = 2$). (h) is the result of FCGF-1 ($\sigma = 0.5, \epsilon = 0.1^4$).

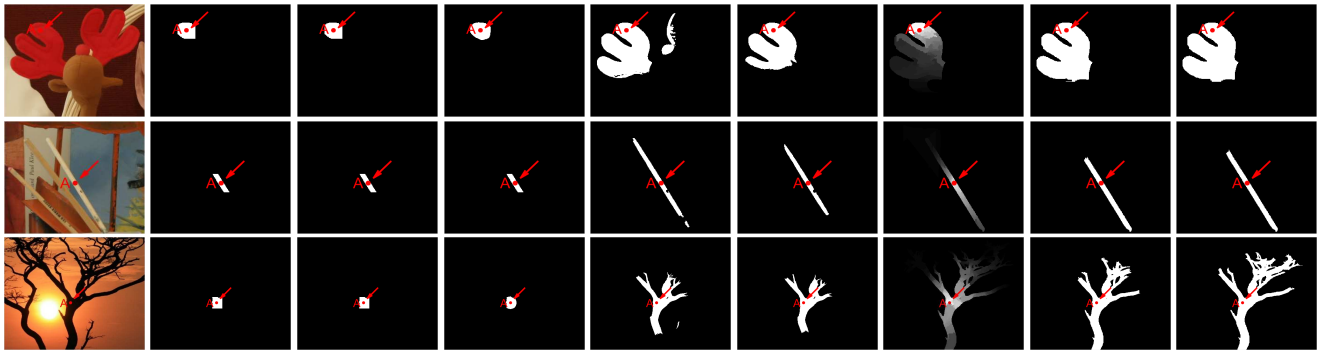
5.2. Distance-aware Weights

The tree distance d_{ij}^t can reflect the Euclidean distance d_{ij}^e. For arbitrary two pixels i and j on a grid, we always have $d_{ij}^t \geq |x_i - x_j| + |y_i - y_j| \geq d_{ij}^e$. Theoretically, the first inequality holds as $|x_i - x_j| + |y_i - y_j|$ is the minimal tree distance between i and j on a grid. The second inequality is just the triangle inequality. Specifically, for two pixels in the same region, d_{ij}^t approaches to d_{ij}^e. For two pixels on either side of an edge, we also have $d_{ij}^t > d_{ij}^e$ as the two pixels are connected by a long path. Indeed, no matter how d_{ij}^t changes, d_{ij}^t is always greater than d_{ij}^e. Therefore the tree similarity w_{ij}^t (10) is a spatial affinity.

Due to w_{ij}^t, our filter kernel W_{ij}^{fc} considers the spatial affinity. We can observe this from Eq.(19) clearly.

$$W_{ij}^{fc} = \frac{\sum_{k\in\Omega}\left(1 + \frac{(I_i - E^w(I))(I_j - E^w(I))}{D^w(I)+\epsilon}\right) w_{ik}^t w_{kj}^t}{(\sum_{k\in\Omega} w_{ik}^t)(\sum_{k\in\Omega} w_{kj}^t)} \tag{19}$$

Compared with the kernel W_{ij}^{gf} (1) of GF, W_{ij}^{fc} contains an extra term w_{ij}^t that measures the spatial-affinity. For more clearly illustration, we visualize six kernels in Fig.2, where only MLPA-2, FCBF, TF and FCGF-1 shrink the kernel's amplitudes with increasing distance and consider the spatial



(a) Input (b) GF (c) CLMF-1 (d) MLPA-2 (e) FCBF (f) TF (g) FCGF-1 (h) FCGF-1 (i) FCGF-1

Figure 4: Support region comparison. (a) is the input image. (b) (c) and (d) are the implicit support regions of GF, CLMF-1 and MLPA-2 in the predefined explicit support regions, where the threshold $T = 0.001$, the maximal arm length or the radius of box window are 10. (e) and (f) are the implicit support regions of FCBF and TF with the same threshold $T = 0.001$. (g) demonstrates the kernel weights of FCGF-1 between A and the rest points in the image domain. (h) and (i) exhibit the implicit support regions composed by the pixels with weights larger than 0.001 and 0.0001, respectively. The shapes of two implicit support regions are almost same and both choose almost all the most relevant points. Compared with (e) (f) (h) and (i), (b) (c) and (d) obviously lose many relevant points.

affinity. Compared with FCGF-1, MLPA-2 is constrained by a local support region and selects few pixels in Fig.2d. Hence, it is inferior to our filter. Compared with FCBF, FCGF-1 attenuates the kernel weights along with the shape of the arrow man. In contrast, FCBF has no idea of the arrow man and simply attenuates its kernel weights according to the Euclidean distance as shown in Fig.2e, where the red arrow indicates the direction of attenuation and the concentric circles denote the isoline of amplitudes. Since TF is implemented by TMF followed by FCBF, the tree distance in TMF is apt to be corrupted by the Euclidean distance in FCBF. Hence the kernel of TF in Fig.2f is similar to the kernel of FCBF in Fig.2e.

The spatial affinity is critical when we perform the fully connected filtering under the guidance of an image which is consisted of several color regions as illustrated in Fig.3a because the intensity similarity fails to distinguish the differences of pixels in the same color region. To illustrate the function of the spatial affinity, we relax the local support regions of GF, CLMF-1 and MLPA-2 to the entire image domain Ω and evaluate filtering results using PSNR, where GF is 10.3dB, CLMF-1 is 10.6dB, MLPA-2 is 17.4dB, TF is 17.7dB and FCGF-1 is 18.2dB. The results of TF and MLPA-2 are almost same because TF considers the spatial affinity and MLPA-2 takes into account a spatial term. The fully connected filtering results of GF and CLMF-1 are overly smoothed due to missing the spatial affinity. TF and MLPA-2's PSNR are similar to FCGF-1, but the running cost of our filter the cheapest according to Table 1.

5.3. Implicit geometric-adaptive support regions

Whether the implicit support region defined by the kernel of a filter can efficiently present complicated support regions determines the filtering performance. In this section we compute all-pairwise-weights on three synthesized pictures, including a concave object image, a slender structure image and a tree image, and select the pixels with weights larger than 0.001 as the implicit support region for comparison. If the weights outside the explicit support region are assumed to be zeros, GF, CLMF and MLPA also have implicit support regions. But the possible pixels composing the implicit support region are constrained by the explicit support region. For comprehensive comparison, we demonstrate the two kinds of implicit support regions in Fig.4. Although CLMF-1 and MLPA-2 determine the support region in data-drive manner, their implicit support regions also lose much information as the implicit support region of GF does because all their implicit support regions are limited by the maximal size of the explicit local support region (*i.e.* the maximal radius of the box window of GF or the maximal arm length of the crossed based support region of CLMF-1 and MLPA-2). Specifically, for the slender structure image, they only select a tiny part of the entire region. For the tree image, the three methods completely lose many branch structures. The implicit support region of FCBF can present the complicated support regions in Fig.4, but FCBF is apt to introduce separated regions to the implicit support region because the Euclidean distance cannot reflect the underlying structure of the guidance image. The tree distance conquers the shortcoming and therefore the support region of FCGF is rather stable for different thresholds as illustrated in Figs.4h 4i. Under the guidance of the tree distance, the implicit support region of TF also removes the disconnected regions. These comparisons prove that the MST based implicit support region is geometric-adaptive and better than the other kinds of support regions.

(a) Guidance (b) Mask (c) GF (d) CLMF-1 (e) MLPA-2 (f) TF (g) FCGF-1

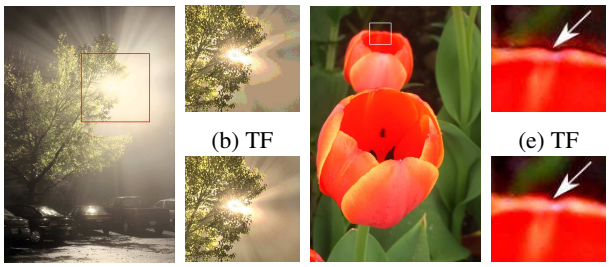
Figure 5: Image matting. A binary mask (b) is filtered under the guidance of (a). We compare FCGF-1 ($\sigma = 0.1, \epsilon = 1$) with GF ($\sigma = 60, \epsilon = 0.1^2$) , CLMF-1 ($r = 60, \tau = 70, \epsilon = 0.1^2$), MPLA-2 ($k = 0.005/255, \epsilon_s = 0.3, \epsilon_r = 0.1^4$), TF ($\sigma = 0.5, \sigma_s = 10, \sigma_r = 0.05$).

6. Experiments and Applications

To achieve different smoothing effects, researchers designed different edge-preserving filters. TF employs the spatial-intensity similarity to perform weighted average. GF adopts an edge-preserving linear model. Both CLMF and MLPA introduce a local geometric-adaptive support region to the edge-preserving linear model. We introduce the tree similarity to the linear ridge regression model and relax the local support region to the entire image domain. In this section, we verify these edge-preserving techniques and demonstrate our filter's advantages on a variety of computer vision and graphics applications.

Image Matting Thanks to the fully connection defined on the MST, our filter is able to preserve the tiny curved tassels structure illustrated in Fig.5a. To show this, we apply GF, CLMF-1, MLPA-2, FCGF-1 and TF on a binary mask and refine it to appear an alpha matte near the object boundaries and illustrate the matting results in Fig.5, where the binary mask is obtained from segmentation and the guidance I is a color image. In order to map the tassels of I into the final result, GF has to choose a large box Ω_i^b which leaks information and blurs the sharp boundaries. As for CLMF-1 and MLPA-2, their support regions fail to represent the curved tassels, thereby the matting results are rather unsatisfying. Unlike these GF-like filters, our filter is able to transfer the tiny tassels from the guidance image to the matting results while preserving sharp edges for the reason that MST does not cross the color/intensity edges of an image and automatically drags away two dissimilar pixels that are close to each other in the spatial domain. TF also takes MST to represent the underlying geometric structure, hence it can produce the similar matting result as our filter. However, compared with our filter, its speed is extremely slow according to Table 1. Specifically, our filter spends 0.6s. In contrast, TF costs nearly 20s in our laptop.

HDR compression and Detail enhancement TF and FCGF have similar behavior in some applications. We can observe this from their implicit support regions in Fig.4 and the matting result in Fig.5 clearly. This is because the two filters are built on the weighted average operator $E^w(x)$ (or equivalently TMF) which takes advantage of MST. The



(a) Input (b) TF (c) FCGF-1 (d) Input (e) TF (f) FCGF-1

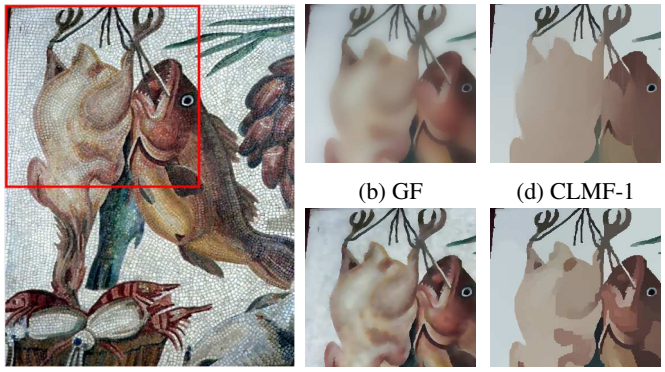
Figure 6: Quantization artifacts in HDR compression and gradient reversal artifacts in detail enhancement. The parameters are ($\sigma = 0.5, \sigma_s = 5, \sigma_r = 0.1$) for TF and ($\sigma = 1, \epsilon = 1$) for FCGF-1.

major difference between them is that FCGF is a GF-like filter and TF is a BF-like filter. Hence TF is likely to inherit the shortcoming of BF. One problem shared by TF and BF is that their brute-force implementations are time-consuming. Bao *et al*. in the paper [1] exploit acceleration techniques [17] to speed up TF, but these methods may have noticeable quantization artifacts in HDR compression [7] as shown in Fig.6b. In contrast, the naive implementation of FCGF can be computed in linear time and thus avoid the quantization artifacts produced by acceleration techniques. Another problem is the gradient reversal artifacts [5, 23] in detail enhancement [6], which has been solved by GF. Since FCGF is a GF-like filter, it does not have this problem. We can observe this in Figs.6e 6f.

Flash/no-Flash Detail Transfer with Denoising A flash image is able to capture the high-frequency texture and detail. On the contrary, a no-flash image captures the overall appearance with noisy. To take the visual richness of a real environment, we use the relatively noise-free flash image to reduce noise in the no-flash image and to transfer high-frequency detail from the flash image to the denoised image. Following the procedure described in [16], we produce the results of GF, CLMF-1, MLPA-2, TF and FCGF-1 and demonstrate them in Fig.7. Compared with other filters, our filter not only takes the advantage of large filter kernels to suppress noises as much as possible but also transfers the detail information in the flash image while removing the shadow regions under the guidance of the flash image. This is because the flash image's structure is represented by an

(a) Flash (b) No-flash (c) GF (d) CLMF-1 (e) MLPA-2 (f) TF (g) FCGF-1

Figure 7: Flash/no-Flash denoising. (a) a flash image. (b) a no-flash image. (c) (d) (e) (f) (g) are the results of GF, CLMF-1, MLPA-2, TF and FCGF-1 with parameter settings ($r = 40, \epsilon = 0.02^2$) for GF, ($r = 9, \tau = 25, \epsilon = 0.1^5$) for CLMF-1, ($k = 0.007/255, \epsilon_s = 0, \epsilon_r = 0.02^2$) for MLPA-2, ($\sigma = 0.3, \sigma_s = 8, \sigma_r = 0.06$) for TF and ($\sigma = 1, \epsilon = 0.1^5$) for FCGF-1.



(a) Input (b) GF (c) TF (d) CLMF-1 (e) FCGF-1

Figure 8: Texture smoothing results produced by 5 iteration of filtering. (a) is the input image. (b) is the result of GF ($r = 5, \epsilon = 0.1^2$). (c) is the result of TF ($\sigma = 0.01, \sigma_s = 2, \sigma_r = 0.05$). (d) the result of CLMF-1 ($r = 25, \tau = 50, \epsilon = 0.1$). (e) is the result of FCGF-1 ($\sigma = 0.5, \epsilon = 1$).

MST and the filtering weight W_{ij}^{fc} is learned from the MST. In contrast, other GF-like filters cannot efficiently represent the underlying structure of the flash image. We can convincingly verify this difference from the A, B and C points in Fig.7. Although TF exploits MST, it cannot behave as our filter. We own this to that the Euclidean distance in FCBF corrupts the tree distance in TMF, which makes the tree similarity fail to represent the underlying geometric structure.

Texture Smoothing Texture smoothing that removes tiny structures while preserving major structures is a challenging edge-preserving smoothing task. Previous GF-like filters distinguish textures from major image structures based on pixel color/intensity differences. We can verify this from their weighted average forms $p_i = \sum_{j \in \Omega_i} W_{ij} q_j$ which only involve intensities. Thanks to the edge-preserving ability of the weighted average form, these filters have found their places in many applications. But, employing these filters to suppress texture is not a wise choice. We can observe this from Fig.8. This is because texture should be distinguished from major structures by their spatial scale, rather than by their contrasts. Hence GF and CLMF-1 cannot discriminate the boundaries of textures from the boundaries of major structures. TF makes a distinction between textures and major image structures based on the connections of MST (major structures contain a big number of connected pixels) and produces a satisfactory texture smoothing result. Similar to TF, FCGF also employs MST and thus has the ability of TF. Following the iterative scheme [25], our filter provides a much better edge-preserving result. This is because TMF, a component of TF, only considers the tree distance and thus suffers from "leak" problem [1] of MST. In contrast, our filter takes into account the spatial affinity and the intensity similarity simultaneously.

The texture smoothing ability does not conflict with the tiny structure preserving ability demonstrated in the image matting application. First, MST can detect and represent arbitrary structures no matter tiny structures and major structures. For tiny structures, the number of connected pixels is very small. In contrast, there are many connected pixels in the major structures. Second, the texture smoothing result is produced by multiple times filtering. The matting result is obtained from one time filtering. Third, since tiny structures are represented by few connected pixels, it is more likely to be smoothed out than major structures via multiple times filtering. This also explains why the intensity based filtering methods fail to complete the task and smooth out both textures and major structures indiscriminately in Figs.8b 8d.

7. Conclusion

In this paper, we propose a fully connected guided filter by introducing an MST to GF. Unlike other GF-like filters, the filtering result takes into account the distance between two pixels without loss of efficiency. More importantly, its implicit support region defined by the large weights automatically can adapt itself according to the underlying structure. Compared with the BF-like TF, our filter can be computed in linear time and avoid quantization artifacts and gradient reversal artifacts. These abilities make our filter outperform previous filters.

Acknowledgement

This work was supported by China 863 program with No. 2013AA013701, and by NSFC with Nos. 61331018, 91338202, 61271430, 61571046.

References

[1] L. Bao, Y. Song, Q. Yang, H. Yuan, and G. Wang. Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree. *Transactions on Image Processing*, 23(2):555–569, Feb 2014. 2, 4, 5, 7, 8

[2] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. 5

[3] L. Dai, F. Zhang, X. Mei, and X. Zhang. Fast minimax path-based joint depth interpolation. *Signal Processing Letters*, 22(5):623–627, May 2015. 4

[4] Y. Ding, J. Xiao, and J. Yu. Importance filtering for image retargeting. In *Conference on Computer Vision and Pattern Recognition*, pages 89–96, June 2011. 1

[5] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 21(3):257–266, July 2002. 7

[6] R. Fattal, M. Agrawala, and S. Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics*, 26(3), July 2007. 7

[7] R. Fattal, D. Lischinski, and M. Werman. Gradient domain high dynamic range compression. *ACM Transactions on Graphics*, 21(3):249–256, July 2002. 7

[8] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, March 2007. 3

[9] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, June 2013. 1, 2, 4

[10] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):504–511, Feb 2013. 1, 2

[11] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola. From local kernel to nonlocal multiple-model image denoising. *International Journal of Computer Vision*, 86(1):1–32, 2010. 2, 4

[12] K.-H. Kim and S. Choi. Walking on minimax paths for k-nn search. In *Conference on Artificial Intelligence*, 2013. 3

[13] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694, Aug. 2004. 1

[14] J. Lu, K. Shi, D. Min, L. Lin, and M. Do. Cross-based local multipoint filtering. In *Conference on Computer Vision and Pattern Recognition*, pages 430–437, June 2012. 1, 2, 4

[15] J. Park, H. Kim, Y.-W. Tai, M. Brown, and I. Kweon. High quality depth map upsampling for 3d-tof cameras. In *International Conference on Computer Vision*, pages 1623–1630, Nov 2011. 2

[16] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3):664–672, Aug. 2004. 7

[17] F. Porikli. Constant time o(1) bilateral filtering. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008. 7

[18] X. Tan, C. Sun, and T. Pham. Multipoint filtering with local polynomial approximation and range guidance. In *Conference on Computer Vision and Pattern Recognition*, pages 2941–2948, June 2014. 1, 2, 4

[19] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision*, pages 839–846, Jan 1998. 1, 2

[20] L. Xu, Q. Yan, and J. Jia. A sparse control model for image and video editing. *ACM Trans. Graph.*, 32(6):197:1–197:10, Nov. 2013. 2

[21] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via relative total variation. *ACM Transactions on Graphics*, 31(6):139:1–139:10, Nov. 2012. 2

[22] Q. Yang. Stereo matching using tree filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(4):834–846, April 2015. 2, 3, 4, 5

[23] Q. Yang, K.-H. Tan, and N. Ahuja. Real-time o(1) bilateral filtering. In *Conference on Computer Vision and Pattern Recognition*, pages 557–564, June 2009. 7

[24] L. Yen, M. Saerens, A. Mantrach, and M. Shimbo. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *Conference on Knowledge Discovery and Data Mining*, pages 785–793, 2008. 3

[25] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *European Conference on Computer Vision*, pages 815–830, 2014. 8