Recursive Edge-Aware Filters for Stereo Matching

Cevahir Çığla
Aselsan Inc.
Ankara/Turkey
ccigla@aselsan.com.tr
www.cevahircigla.com

Abstract

In this study, taxonomy of recursive edge-aware filters (REAF) is provided, with the introduction of new approaches to the state-of-the-art. The one tap recursive filters are classified according to recursion rate calculation, recursion type and the unification of reverse directions. In that manner, eight types of edge-aware recursive filters are defined, where only three of them are addressed in literature so far. Comprehensive analyses are provided based on computational complexity and filter characteristics which affect the use of such filters for various applications. In order to compare the capabilities of these filters, stereo matching, as one of the most common application area of edge-aware filters, is considered and extensive experiments are provided through well known datasets. The evaluation is conducted through large number of stereo pairs on both CPU and GPU with independent parameter optimization of each filter that provides fair comparison. According to the experimental results, advantages of un-normalized recursion for matching accuracy and sequential integration of reverse directions for execution speed are illustrated as important conclusions for future directions of REAFs.

1. Introduction

In recent years, edge-aware filters [1]-[6] have been popular for various applications in computer vision. The fundamental idea behind this type of filtering is to provide local adaptive weights within a pre-defined window by highlighting color-wise similar pixels and achieving weighted averaging. Recently, the bilateral filter (BF) [1] and guided filter (GF) [2] have become the most popular of such filters with wide application areas [7]-[13].

Despite the advantages, definition of window size and use of specified integral images for speeding-up, decrease adaptability of these algorithms for handling regions with various local color distributions. In this manner, they are not completely adaptive to image content. Besides, computational complexity is still high and suppressive for real-time applications on various platforms. At that point, recursive filters [3]-[6] that have been introduced recently address the problem of window size dependence and computational complexity for efficient edge-aware filtering. Moreover, recursive filters enable connected support regions that constrain geometrical relevance among the neighbor pixels. This is important especially for the geometry related applications such as stereo matching, motion estimation and segmentation.

In this paper, recently proposed recursive edge-aware filters (REAF) are profoundly analyzed and classified according to their recursion characteristics. Moreover, a generalization is provided that introduces new approaches to state-of-the-art. All types of one-tap recursive filters are compared in terms of edge-preserving characteristics as well as performance through one of the most common and widely addressed application, stereo matching.

In section 2, a brief introduction to REAFs is provided which is followed by the generalization of REAFs. Section 3 is devoted to the analyses of REAFs in terms of complexity and filter characteristics. In Section 4, experimental results are presented based on the performances in stereo matching as the most common application area of REAFs, Finally the paper concludes in Section 5.

2. Recursive Edge-Aware Filters

The relation between input, x, and output, y, 1-D signals in an n-th order recursive system is given by the following formulae;

$$\vec{y}_i = \sum_{k=i-1-n}^{i} \beta_k x_k + \sum_{k=i-n}^{i-1} \alpha_k \vec{y}_k \qquad (1)$$

where α and β denote the weight coefficients of the corresponding values. The causal system given in (1) can be extended, especially for image signals by the anti-causal part, where the reverse direction is traversed with the same order. Finally, the output is obtained as a combination of causal and anti-causal outcomes. The 1-D signals can be considered as the scan lines of an image corresponding to rows and columns; though 2D filtering of an image is achieved by performing recursive operations along 1-D horizontal and vertical axes, as in steerable filters [14],

Considering state-of-the-art REAFs [3]-[6] applied for images, the most common type of recursion is 1st order

which balances accuracy and computational complexity. In the 1st order recursion, relations can be simplified as

$$\vec{y}_i = \beta_i x_i + \alpha_{i-1}\,\vec{y}_{i-1}$$
$$\overleftarrow{y}_i = \beta_i x_i + \alpha_{i+1}\,\overleftarrow{y}_{i+1}. \quad (2)$$

where weight coefficients correspond to the recursion rates of pixels among scan order. Originating from the same idea of successive updates along orthogonal axes; [3]-[6] involve differences in the update rules (relation of coefficients) and unification of reverse directions. In *Permeability Filter* (PF) [6], designed for cost aggregation in stereo matching, output of the recursive operation among one direction is achieved through un-normalized weighting by setting β_i=1. In *Bi-exponential Edge-preserving Smoother* (BEEPS) [3], *Recursive Bilateral Filter* (RBF) [4] and *Domain Transform* (DT) [5] the output is achieved by setting β_i as $(1-\alpha_{i-1})$ and $(1-\alpha_{i+1})$ correspondingly. There are also variations in calculation of recursion rates as well as unification of reverse directions which are discussed in the following sub-sections.

2.1. Recursion Rate Calculation

In general, a guidance image (GI) is defined as the source of weights that are exploited to filter data for edge-aware filters. In 1st order REAFs [3]-[6], recursion rates are calculated according to the similarities between neighboring (consecutive) pixels along the scan direction. As the common approach, each technique in state-of-the-art utilizes Gaussian weighting function in order to map RGB or intensity differences of consecutive pixels to the range of recursion rates as follows:

$$\alpha_i = f\left(\left|x_i - x_{i-1}\right| / \sigma\right), \qquad f(x) = e^{-x} \quad (3)$$

where σ denotes the smoothing factor. In PF, RBF and DT, recursion rates are calculated through the guidance image (*GI*) by considering intensity similarity between neighboring pixels (X_i and X_{i-1}), on the other hand in BEEPS, α_i is determined by the similarity between guidance image and its filtered version (GIF) through Y_i and X_{i-1}.

2.2. Recursion Type

There are two choices for the recursive update in 1st order filters according to the distribution of update rates between two neighboring pixels, normalized and un-normalized. This has a significant importance on the relation (effective weight) between distant pixels, which is the multiplication of weights within. In order to analyze the effect of recursion, consider the toy 1D example given in Figure 1.

Figure 1: The pixel-wise relations along a row/column.

The output for the normalized recursion along the left-to-right scan is achieved as,

$$\vec{Y}_1 = (1-\alpha_0)X_1 + \alpha_0 X_0$$
$$\vec{Y}_2 = (1-\alpha_1)X_2 + \alpha_1\left[(1-\alpha_0)X_1 + \alpha_0 X_0\right] \quad (4)$$

In this case, valid for DT, RBF and BEEPS, the effective weight, W_{ij}, between two pixels, with indexes *i* and *j*, changes according to the scan direction with a damping factor of the normalizing rate as follows:

$$W_{ij} = (1-\alpha_i)\prod_{p=i+1}^{j}\alpha_p,\; W_{ji} = (1-\alpha_j)\prod_{p=i}^{j-1}\alpha_p \quad (5)$$

On the other hand, the update rule of PF provides an un-normalized recursion as,

$$\vec{Y}_1 = X_1 + \alpha_0 X_0$$
$$\vec{Y}_2 = X_2 + \alpha_1\left[X_1 + \alpha_0 X_0\right] \quad (6)$$

yielding an effective symmetric weight between two pixels as the successive multiplication of recursion rates:

$$W_{ji} = W_{ij} = \prod_{p=i}^{j}\alpha_p \quad (7)$$

It is clear that normalized recursion does not provide symmetry between two pixels, besides the recursion effect is decreased faster due to (1-α) damping. This damping effect violates connectedness of the aggregation region such that neighboring pixels with same intensity (with update rates of α close to 1) even cannot support each other while distant pixels may have higher contribution. This is especially problematic for cost aggregation in geometric applications where support of nearby similar pixels is strongly desired. On the other hand, un-normalized recursion not only preserves symmetry but also yields geometrically more reliable support without any damping.

2.3. Integration of Recursions

The final important step is the integration of recursive operations in opposite directions. There are two techniques for integration; sequential approach (DT and RBF), performs recursions consequently where the second recursion is executed on the output of the first recursion. In independent integration (BEEPS and PF), recursions along reverse directions are performed independently on the original input and the final output is obtained by the summation of two outcomes.

The weight distribution for the independent recursion is same as the relation given in (5) and (7), since addition of uncorrelated data from the reverse directions is conducted. However, this is not valid for sequential recursion since output of the first recursion has an additional influence in

the second recursion. Let us consider the effect of X_0 on X_2 in Figure 1, where the left-to-right recursion is followed by the right-to-left recursion. In the first recursion, effect of X_0, indicated as W_{02}, is given by (5) and (7) depending on the recursion type. We can consider only the normalized case for the sake of simplicity. It is obvious that X_0 has influence on the proceeding pixels of X_2, which are the sequentially multiplied versions of W_{02} as illustrated in Figure 2. So, the effect of X_0 on X_2 continues in the second recursion (red arrows) as a reflection effect. This is also valid for all the preceding pixels (on the left) of X_2.

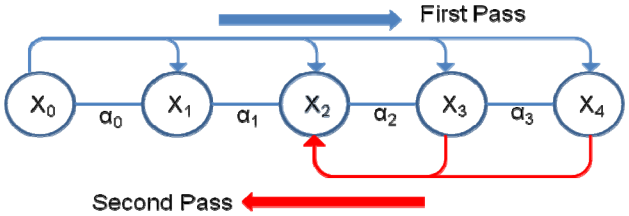

Figure 2: The reflection effect of X_0 in the second recursion

In a generalized form, the reflection effect of a pixel, *m*, on the target pixel *n*, can be defined according to its position as follows:

$$R_{mn} = \begin{cases} \sum_{k=n}^{N-1}(1-\alpha_{k+1})\prod_{j=n}^{k}\alpha_j^2 & m < n \\ \sum_{k=m}^{N-1}(1-\alpha_{k+1})\prod_{j=m}^{k}\alpha_j^2 & m > n \end{cases} \quad (8)$$

where *N* is the length of the row. The reflection effects of preceding pixels are independent of their spatial distance to target, *n*, and fixed for all. On the other hand, reflection of the proceeding pixels depends on the spatial distance. As the pixels get away from the target pixel, *n*, reflection effect is decreased. Considering the effect of reflection, closed form of the weight between two pixels after sequential recursion is determined as:

$$W_{mn} = \begin{cases} [1+R_{mn}](1-\alpha_{m-1})\prod_{j=m}^{n-1}\alpha_j & m < n \\ [1+R_{mn}](1-\alpha_{m-1})\prod_{j=n}^{m-1}\alpha_j & m > n \end{cases} \quad (9)$$

In (9), the first term corresponds to direct effect and the second term corresponds to reflection. It is clear that, in sequential case, preceding pixels in the first recursion have more influence due to the reflection, hence symmetry is lost. Meanwhile, independent recursion is balanced between reverse directions.

Considering the variation of rate calculation, recursion type and integration of recursion, where each case involve two types within, 1st order REAFs can be classified into eight classes as given in Table 1. Actually, this classification is also valid regardless of the REAF order. The REAFs introduced in state-of-the-art only covers three types in Table 1; PF corresponds to *type1*, RBF-DT corresponds to *type2* and BEEPS corresponds to *type7*.

Table 1: The taxonomy of REAFs

REAF Types	Rate Calculation	Recursion Type	Recursion Integration
Type0	GI	Un-normalized	Sequential
*Type1**	GI	Un-normalized	Independent
*Type2**	GI	Normalized	Sequential
Type3	GI	Normalized	Independent
Type4	GIF	Un-normalized	Sequential
Type5	GIF	Un-normalized	Independent
Type6	GIF	Normalized	Sequential
*Type7**	GIF	Normalized	Independent

3. Analyses of REAFs

In this section, the categorizations of REAFs are analyzed in terms of complexity and filter characteristics based on 2D filtering achieved through horizontal and then vertical 1st order recursions.

3.1. Complexity

Considering the recursive operations, all REAF types exploit successive memory access pattern which is advantageous for prompt operation. The unit operation (defined as the recursion in one direction) in un-normalized recursion involves 1 addition and 1 multiplication along one pass according to modified version of (2), on the other hand normalized recursion involves 2 additions and 1 multiplication. Considering the integration part, the number of operations is given in Table 2 for one pass 2D filtering and it is clear that the most efficient types are *type0* and *type4* with un-normalized recursion and sequential integration of reverse directions. The difference in memory requirement between REAF types is negligible, where independent integration requires additional one row/column memory.

Table 2: Complexity depending on recursion/integration types

# of ADDs / MULs	Un-Normalized	Normalized
Sequential	4 / 4	8 / 4
Independent	6 / 4	10 / 4

3.2. Filter Characteristics

As mentioned in section 2, recursion type and recursion integration have significant influence on the effective filter through variation on distribution of weights. According to discussion in section 2, normalized recursion yield non-symmetric disjoint support regions, while un-normalized recursion is more balanced with maximal use of neighboring relations. The symmetry is preserved with

independent integration, while sequential integration introduces bias for the preceding recursion.

Even though the theoretical analyses of the effect of recursion rate calculation is much more complicated, its effect is not ignorable. Actually, calculation of recursion rates through the comparison of input and filtered data, i.e. GIF, results in four different rate distribution valid for *type4*, *type5*, *type6* and *type7* while same rates are utilized for the first four type in which only the input data and its neighboring relations are considered.

In order to clarify the variation of EAF characteristics, effective weight distributions of the center pixels on different patches are illustrated for eight types in Figure 3. For the sake of visibility, darker regions correspond to higher effective weights. It is important to note that; weight distributions are provided through same σ for all REAFs. According to Figure 3, it is clear that types involving un-normalized recursion yield much wider support among the color-wise similar neighboring pixels. The sequential integration of reverse directions increases the influence of pixels on upper-left of the center pixel since left-to-right and top-to-bottom directions are exploited as initial order.

4. Experiments

In this study, performance of the discussed REAFs is measured through wide range of stereo datasets for the aim of stereo matching, as the most common application. In this manner, a typical local stereo matching approach is followed by cost calculation, aggregation, minimization and occlusion handling steps. To be fair, only the aggregation step is altered by use of eight different types of REAFs, and the other steps are kept constant.

In this setup, extended stereo pairs provided by Middlebury online stereo benchmark [15][16], the new version Middlebury pairs (with its half resolution version as well) and the recent KITTI evaluation benchmark [17] are utilized, whose specifications are given in Table 3. The datasets involve ground truth disparity maps as well, thus estimated disparity maps can be compared with specified error thresholds, such as ratio of pixels with 1-2 level disparity differences w.r.t. ground truth. The experiments are conducted on i5 3.10 GHz 32-bit CPU with 4GB RAM plus Nvidia Quadro4000 graphics card and the code is available upon request.

In stereo matching, pixel similarities are measured by a common approach unifying Census Transform and absolute difference. For each dataset, best cost function is determined by arranging the Census Transform window size which is related to image resolution, and kept constant for each filter. The disparity assignment is achieved by winner take all optimization and the occlusion handling is achieved through left-right consistency check followed by background copying. It is important to note that, for each type of REAFs, parameter optimization, the best choice of σ in (4), is performed independently.

Table 3: The specifications of the benchmark stereo datasets

Specs	Middlebury-1	Middlebury-2	KITTI
Resolution	700 x 550	2800 x 1900	1230 x 370
Disparity	75	270	120
Image No	30	15	190

4.1. Accuracy

Overall performances of REAFs for different datasets are given in Figure 4. The percentage of pixels with 1 level difference in the un-occluded regions is considered for Middlebury-1, KITTI and Middlebury-2 (Half Res.) datasets, and 2 level differences for Middlebury-2 due to larger disparity range. The effect of smoothing parameter (σ) is also given. It is clear that, the best three methods in each dataset are *type1*, *type0* and *type5* consequently that utilize un-normalized recursion. There is %2-3 performance difference on the percentage of erroneous pixels between the normalized (*type2-3-6-7*) and un-normalized recursions (*type0-1-4-5*). On the other hand, the improvement drops to %0.5 when independent integration (even indexed types) is preferred instead of sequential. Thus, use of un-normalized recursion with independent integration obviously provides the best accuracy. The same results are also observed in the average disparity errors given in Table 4, where type1-0 and 5 dominate the best three performances where the rankings are indicated by the super-scripts.

The estimated disparity maps are illustrated with the erroneous pixels along visible (red) and occluded (light blue) regions in Figure 5. The percentages of erroneous pixels are also given on top-left corner of the images. In Figure 5, first three columns belong to stereo pairs taken from Middlebury-1, while the last two columns are devoted for two pairs from Middlebury-2. It is clear that, especially for un-textured surfaces, un-normalized recursion yields more precise disparity estimation, as shown in column three, *type0-1-4-5* model the background smoother.

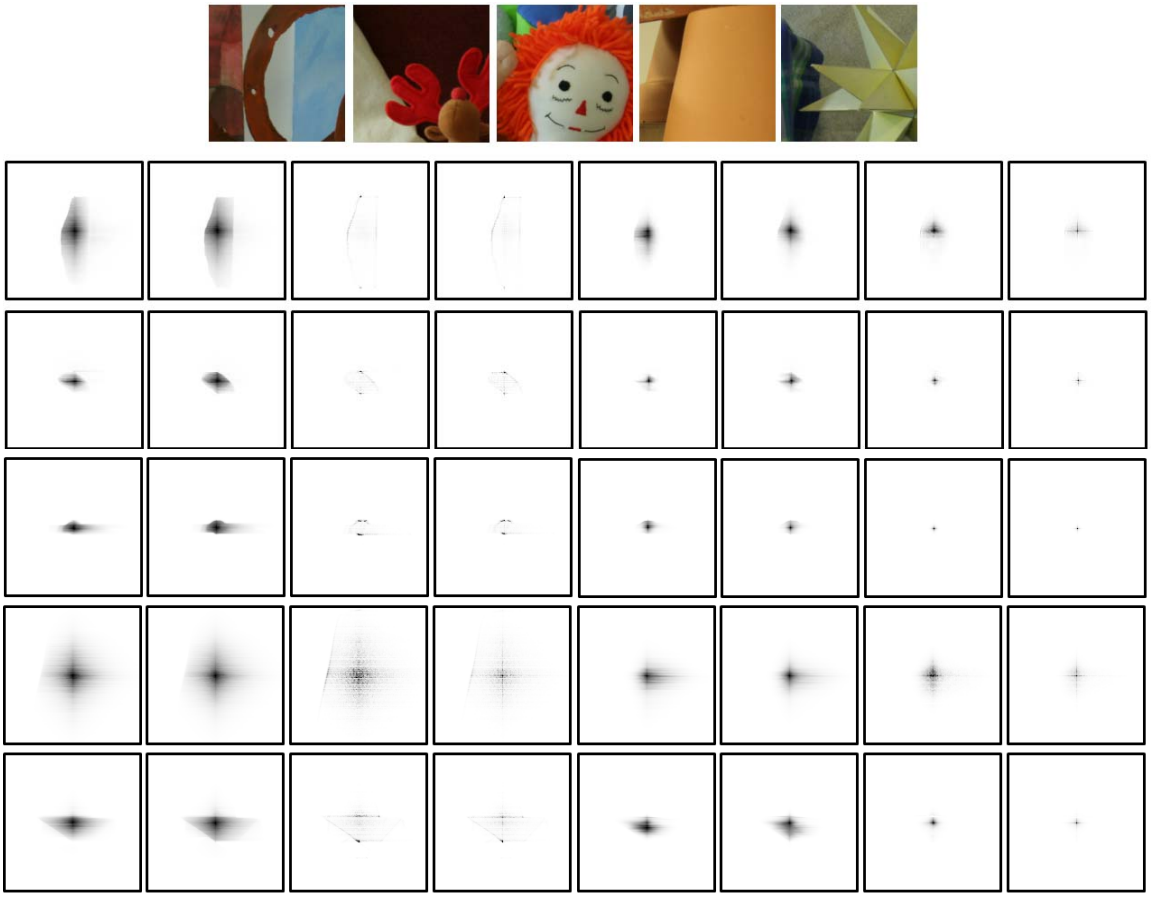

Figure 3: Effective weight distribution of center pixel on different patches given in first row for eight REAF types on each column correspondingly.

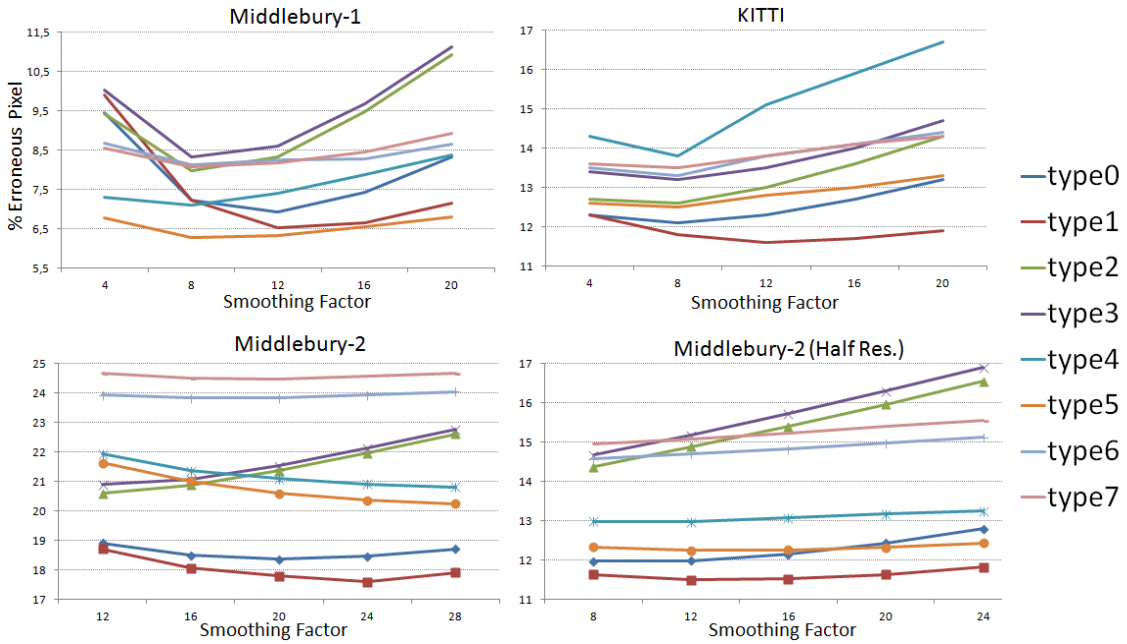

Figure 4: The performance of eight REAFs on four different datasets w.r.t smoothing factor

The results of real-world image database, KITTI, are illustrated in Figure 6. The ground truth provided in this dataset is sparse due to the output of laser scanner, therefore erroneous pixel distribution is also sparse and only the errors along visible pixels are presented. According to the visual results, the out-performance of *type1* is obvious with decreased error rates compared to the other types.

4.2. Computational Complexity

Computation complexity is the other criteria for a preferable cost aggregation methodology. So far, it has been proven by [4][8][9][10] that REAFs are the fastest techniques to filter cost data. In this study, the computational complexity is measured in terms of million pixel aggregation (MPA) per second, which is scalable to different resolution and disparity range. The measured MPAs are given in Table 5 for CPU and GPU, with the rankings indicated by super-scripts. The GPU implementations yield almost x7 speed-up compared to CPU. This is rather a low speed-up compared to common applications in CPU-GPU conversion, which is actually due to the problem of parallelization of recursive operations. The results presented in Table 5 indicates that sequential integration (even indexed types) is faster than the independent integration in CPU due to low number of operation (2adds less for 2D filter) and no use of memory copy. On the other hand, it is slower for GPU since synchronization of threads is required to yield sequential operations. The un-normalized recursion is obviously faster than the normalized counterpart that is consistent with the number of operations given in Table 2. Finally, the effect of rate calculation is insignificant for both CPU and GPU.

Table 4: The Average disparity errors on four different datasets

Average Disp. Error	Middlebury-1	Middlebury-2 / Half Res.	KITTI
Type0	0.91	[2]5.11 / [1]**2.43**	[2]2.24
Type1	[2]0.81	[1]**5.00** / [2]2.46	[1]**2.15**
Type2	1.15	6.58 / 3.22	[3]2.25
Type3	1.21	6.82 / 3.35	2.43
Type4	[3]0.84	5.98 / 2.61	2.54
Type5	[1]**0.76**	[3]5.93 / [3]2.54	2.33
Type6	0.99	7.29 / 3.08	2.53
Type7	1.02	8.11 / 3.15	2.63

According to the experimental results, it is clear that *type1* provides the most precise disparity maps and *type4/type5* yields the fastest computation for CPU/GPU correspondingly. On the other hand, *type0* balances the accuracy and execution speed in CPU, providing a good option among all types; whereas, *type1* seems to be the best choice for GPU implementations. The common feature of the best types in CPU and GPU is the un-normalized recursion yielding accurate estimates through prompt execution. This is an important conclusion; such that there is significant improvement for stereo matching in terms of accuracy as well as complexity when un-normalized recursion is exploited.

Table 5: CPU and GPU computation capabilities of REAF types

Computation MPA/sec	CPU	GPU
Type0	[2]94.07	623.25
Type1	80.19	[2]636.05
Type2	76.49	602.25
Type3	67.73	601.13
Type4	[1]**97.99**	[3]624.10
Type5	[3]81.29	[1]**636.75**
Type6	76.89	603.38
Type7	69.82	602.25

5. Conclusion

In this study, taxonomy of recursive edge-aware filters is provided based on recursion type, integration of reverse directions and rate calculation. New approaches for REAFs are presented with comprehensive analyses through computational complexity and filter characteristics. REAFs are applied for stereo matching and extensive experiments are provided through well known datasets. The experimental results clearly illustrate the advantages of un-normalized recursion for matching accuracy where geometrically consistent support regions are provided. The best performances in terms of execution speed on CPU/GPU are also observed via un-normalized recursion through sequential/independent integration of reverse directions correspondingly, which are crucial conclusions for future directions of REAFs.

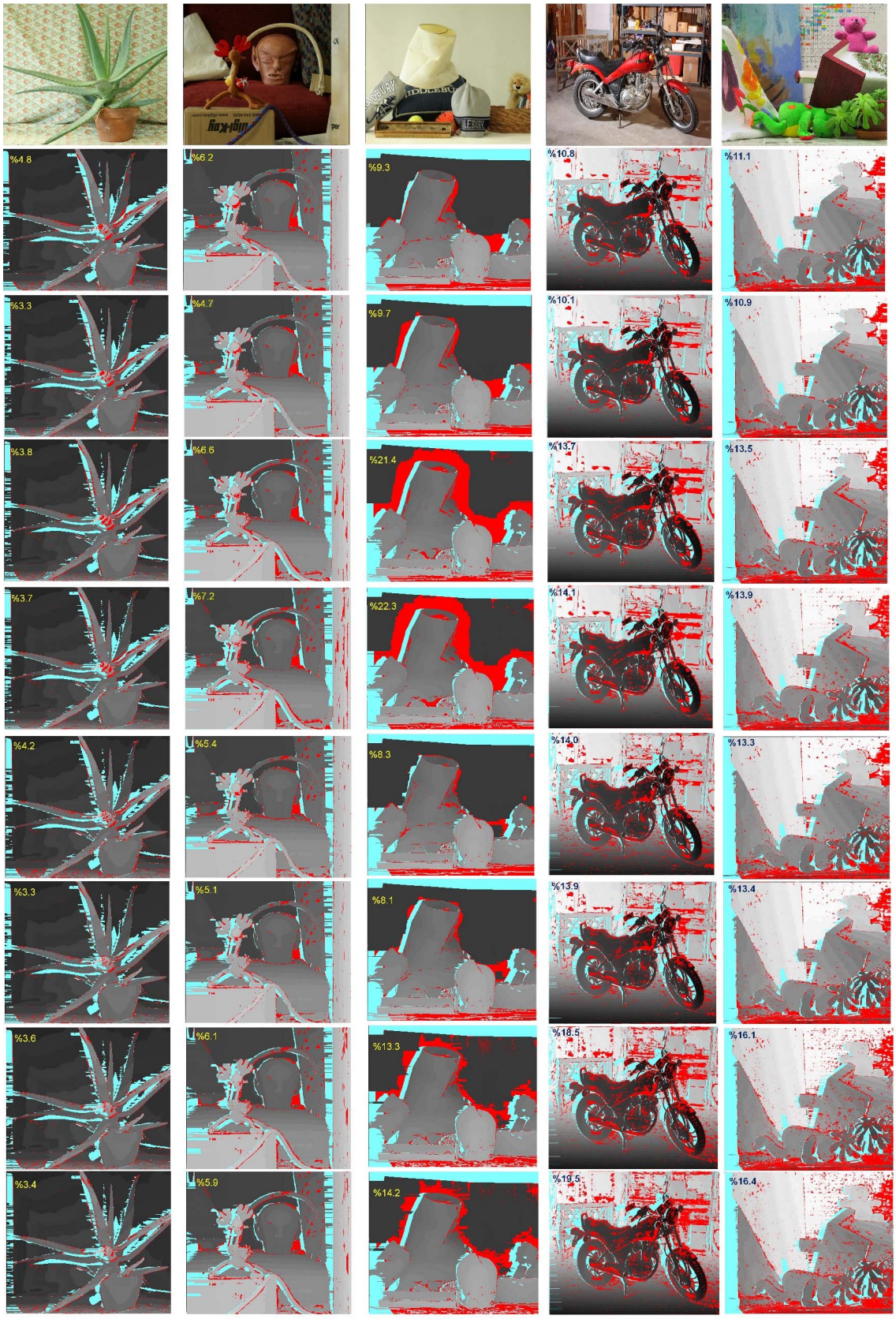

Figure 5: Disparity estimates and errors for type0 to type 7 are given along the rows 2 to rows 9 correspondingly.

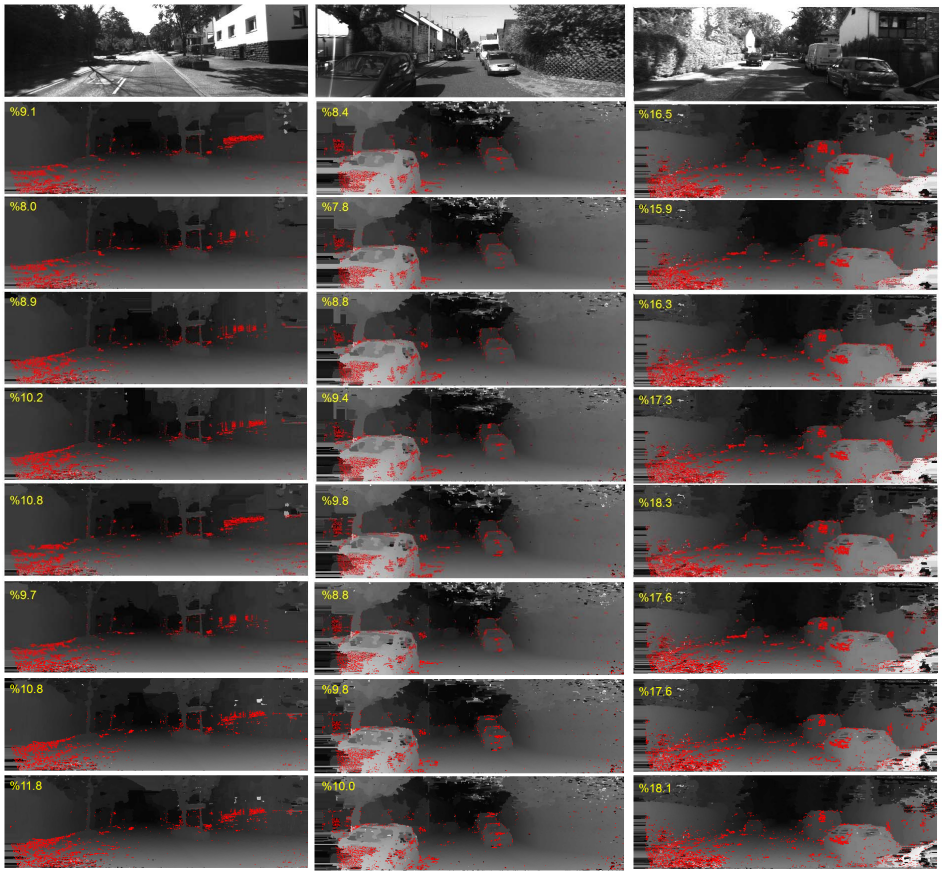

Figure 6: Disparity estimates and errors for type0 to type 7 are given along the rows 2 to rows 9 correspondingly.

References

[1] Tomassi, C., Manduchi, R. *Bilateral Filtering for Gray and Color Images,* In ICCV (1998), 839-846

[2] K. He, J. Sun, and X. Tang, *Guided Image Filtering*, In ECCV, 2010.

[3] Philippe Thévenaz, Daniel Sage, and Michael Unser, *Bi-Exponential Edge-Preserving Smoother*, IEEE TIP, Vol.21 No.9, September 2012

[4] Qingxiong Y., *Recursive Bilateral Filtering*, ECCV, October 2012

[5] E. Gastal and M. Oliveira, *Domain Transform for Edge-Aware Image and Video Processing*, ACM Transactions on Graphics,vol. 30, no. 4, pp. 1-11, July 2011

[6] C. Cigla and A.A. Alatan, *Information Permeability for Stereo Matching*, in Elsevier Journal of Signal Processing and Image Communication, 2013

[7] Yoon, K.J., Kweon, I.S. *Adaptive Support-Weight Approach for Correspondence Search,* IEEE TPAMI (2006)650–656

[8] C. Çığla and A.A.Alatan, *Efficient Edge-Preserving Stereo Matching***,** in ICCV Workshop on Live Dense Reconstruction from Moving Cameras, 2011.

[9] X. Sun, X. Mei, S. Jiao, M.Zhou, Z. Liu and H. Wang, *Real-time Local Stereo via Edge-Aware Disparity Propagation,* Elsevier Pattern Recognition Letters, 2014

[10] C. Pham, J. W. Jeon, *Domain Transformation based Efficient Cost Aggregation for Stereo Matching,* IEEE TCSVT, 2012

[11] Winnemoller, H., Olsen, S.C., Gooch, B. *Real-time Video Abstraction,* In: Siggraph. Volume 25. (2006) 1221–1226

[12] Xiao, J., Cheng, H., Sawhney, H., Rao, C., Isnardi, M. *Bilateral Filtering-Based Optical Flow Estimation with Occlusion Detection*, In: ECCV. (2006)

[13] Yang, Q., Yang, R., Davis, J., Nister, D. *Spatial-Depth Super Resolution for Range Images*, In: CVPR. (2007)

[14] William T. Freeman and EdwardH. Adelson, *The Design and Use of Steerable Filters*, IEEE Transactions on PAMI, September 1991.

[15] D. Sharstein and R. Szeliski, *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*, IJCV Volume 47, April 2002

[16] vision.middlebury.edu/stereo/data/

[17] Geiger A., Lenz P., Urtasun R., *Are We Ready for Autonomous Driving,* CVPR 2012