Digital Image Processing Proposal

Dated: 27.09.2019

Team: Lumos

Project Title: 100+ Times Faster Weighted Median Filter

Project Id: 22

Team Members	TA Assigned
Gulshan Kumar, 20161082	Aditya Aggrawal
Nikhil Bansal, 20161065	

Github Link: https://github.com/gulshan-mittal/Fast-Weighted-Median-Filter

Goals of the Project

Primary Goals

- Perform weighted median filtering of a given image in an efficient manner
- The time complexity of original weighted median filtering (WMF) is $O(n^2)$, where r is the size of the kernel. In our project, we will reduce this complexity to O(n) making it 100+ times faster than the general method.
- To performing the task various Data Structures and mapping will be used which are described later in the proposal.
- The running time of WMF in our project will be shortened from several minutes to less than a second !!

Stretch Goals

- Showing different applications of Weighted median filtering

Problem Statement

Weighted median, in the form of either solver or filter, has been employed in a wide range of computer vision applications for its beneficial properties in sparsity representation.

As a local operator, the weighted median can effectively filter images while not strongly blurring edges. More importantly, it mathematically corresponds to global optimization, which produces results with fully explainable connections to global energy minimization. In many computer vision problems, including stereo matching, optical flow estimation, and image denoising. Making it efficient will have a direct impact on numerous applications using it.

There are many methods which speed up Unweighted median filtering, weighted average (like Bilateral Filtering), etc. but unfortunately methods deployed in these cannot be used here because of the following reasons:

1. The filtering kernel is not separable.
2. It cannot be approximated by interpolation or down-sampling.
3. There is no iterative solution

In this project, we will implement a few algorithms and data structures to perform WMF.

Techniques

The project will have **three major techniques** which include data-weight distribution by a new joint histogram to dynamically allocate weights and pixels, a median tracking algorithm to reduce the time of seeking median by leveraging color coherency of images, and data structure sparsity to reduce the data access cost.

Joint Histogram

A joint histogram is a 2D histogram structure for storing pixel count. Each row corresponds to the feature index and each column corresponds to the intensity value of the pixel and each Histogram block stores the count of pixels corresponding to that intensity value and feature.

In our 2D joint-histogram H, pixel q is put into the histogram bin H(i,f) in the f-th row and i-th column. Thus, the whole joint-histogram is constructed as:

$$H(i,f) = \#\{q \in \mathcal{R}(p) | I(q) = I_i, \mathbf{f}(q) = \mathbf{f}_f\}$$

This counting scheme enables fast weight computation even when the window shifts. Now, for any pixel belonging to the bin (i,f), considering filter center p, its weight can be immediately computed as g(ff,f(p)). By traversing the joint-histogram, all weights can be obtained (see Fig. 1(c)). The total weight for all pixels with value index i is

$$w_i = \sum_{f=0}^{N_f - 1} H(i,f) g(\mathbf{f}_f, \mathbf{f}(p)).$$

The below diagram clearly explains Joint Histogram:

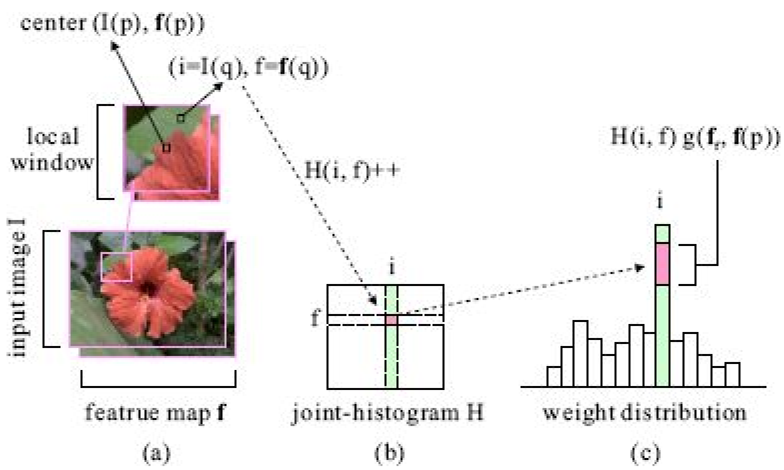


Figure 1.
Joint-histogram illustration. (a) The input image and a local window. (b) Joint-histogram containing all pixels in the window according to value index *i* and feature *f* (c) Total weight calculated for each *i*

For updating, we will be removing the leaving pixels from blocks and add the entering pixels.

Median Tracking

Second data structure used will be median tracking. The main purpose of this is to find the median in an efficient way without iterating over the joint histogram.

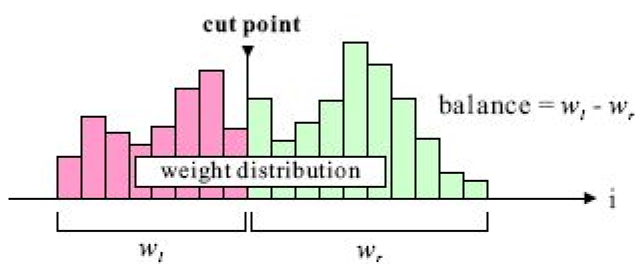


Figure 2. Illustration of the balance and cut point.

For median tracking, we define two terms: Cut Point and Balance where the cut point is a point where cumulative sum reaches half of the total sum and balance is absolute min difference between the left weight and right weight.

Using the property of domain consistency is most images we can easily compute the cut point of the next cut point using the previous cut point. But now we need an efficient way to calculate Balance, for this we define balance computation box (**BCB**) as

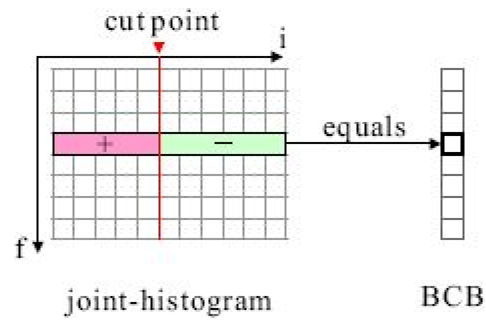


Figure 3. Relationship between BCB and our joint-histogram. By separating each row into the left and right parts according to the cut point in the joint-histogram, their total count difference is put into the corresponding cell in BCB.

As shown in figure it stores the difference of pixel numbers on the two sides of the cut point in the corresponding row of the joint-histogram.

Mathematically

$$B(f) = \#\{q \in \mathcal{R}(p) | I(q) <= c, \mathbf{f}(q) = \mathbf{f}_f\} - \#\{r \in \mathcal{R}(p) | I(r) > c, \mathbf{f}(r) = \mathbf{f}_f\}$$

Using BCB balance can be calculated using

$$b = \sum_{f=0}^{N_f - 1} B(f) g(\mathbf{f}_f, \mathbf{f}(p))$$

Necklace Table

The last Data structure used will be a necklace table with the sole purpose of exploiting sparsity in data distributions of WMF. Necklace Table is a combination of a counting array and a necklace (an unordered doubly linked list).

Unlike traditional linked-list that keeps order for data access, our necklace is unordered thanks to the available array. It allows for very fast traversal by skipping all empty cells. This data structure is used to implement our BCB and all columns of the joint-histogram. It further accelerates element traversal for 10-50 times depending on data sparsity.

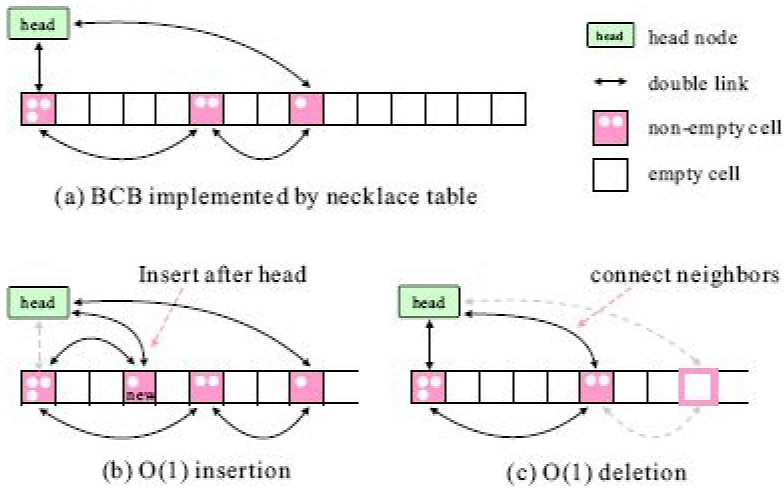


Figure 4. Demonstration of the necklace table.

References

- https://ieeexplore.ieee.org/document/6909758
- http://www.cse.cuhk.edu.hk/~leojia/projects/fastwmedian/index.htm
- Y. Li and S. Osher. A new median formula with applications to PDE based denoising. Commun. Math. Sci, 7(3):741–753, 2009.
- D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010.
- L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L0 gradient minimization. ACM Trans. Graph., 30(6), 2011.
- C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.

Performance Evaluation & Results

After project completion, we will be able to apply weighted median filtering on any data with great efficiency and minimal effect on output.

In the research paper, experiments are conducted on a PC with an Intel i7 3.4GHz CPU and 8GB memory. Only a single thread is used without involving any SIMD instructions.

Table 1 shows the execution time of our algorithm, which is almost linear to the window and image sizes.

Image\Window	10 × 10	50 × 50	100 × 100
320 × 240	0.036s	0.078s	0.114s
640 × 480	0.095s	0.208s	0.331s
1024 × 768	0.223s	0.462s	0.724s
1920 × 1080	0.450s	1.013s	1.667s

Table 1. Execution time with respect to different image sizes and window sizes. The input is an 8-bit single-channel image and the feature is its intensity, i.e., $N_I = 255$ and $N_F = 256$

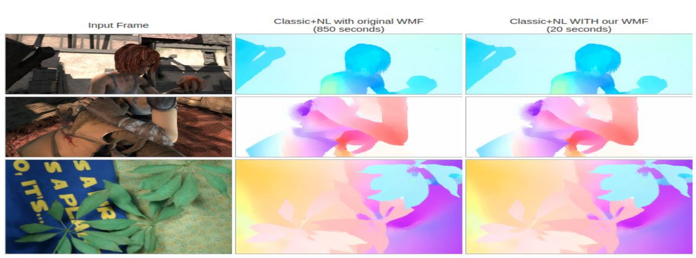
	Step 1	Step 2	Time Saving
Joint	156.9s	0.4s	-
Joint + MT	2.2s	0.5s	98.28%
Joint + NT	3.2s	0.5s	97.65%
Joint + MT + NT	0.3s	0.6s	99.43%

Table 2**. The efficiency of the median tracking and necklace table. We filter one-megapixel RGB images with a 20×20 kernel. Joint, MT, and NT are shorts for joint-histogram, median tracking, and necklace table**

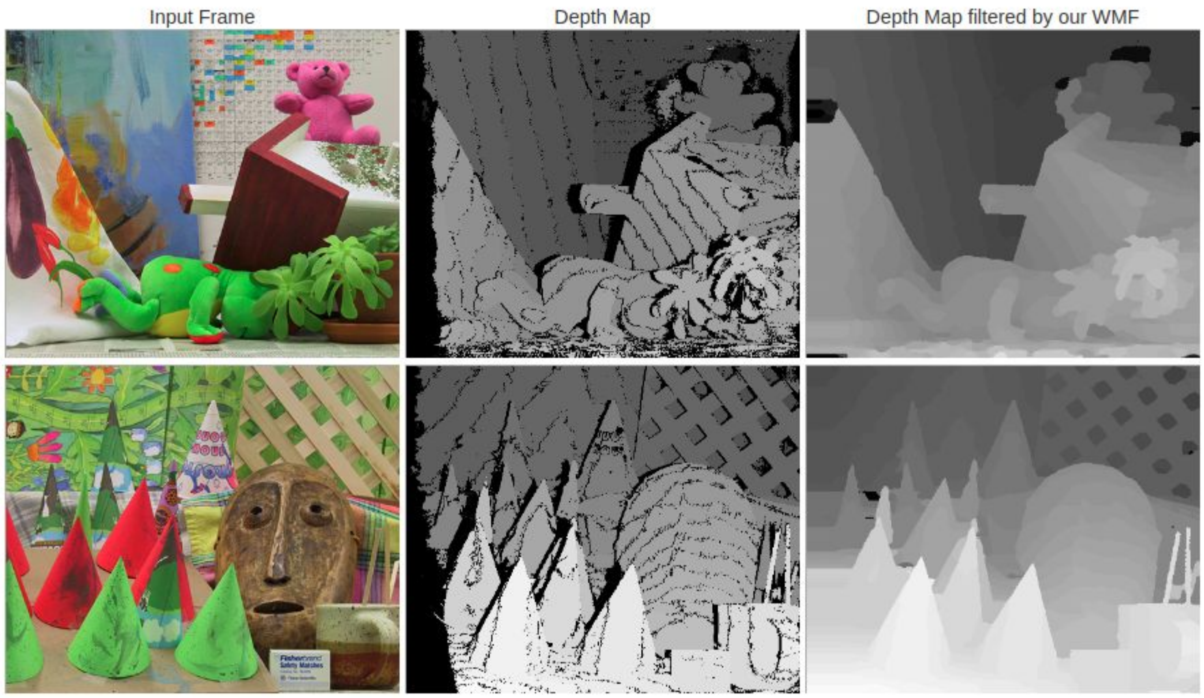
Applications of Weighted Median Filter

Our fast weighted median filter has various applications in many computer vision and image processing topics, such as edge-aware image filtering, guided filtering in optical flow and stereo matching, removing JPEG artifacts and texture smoothing. Other applications of our filter include: solving the sparse-norm optimization problem, cross image filtering, detail enhancement, and tone-mapping, non-photorealistic rendering, guided order filtering and guided sparse-norm filtering.

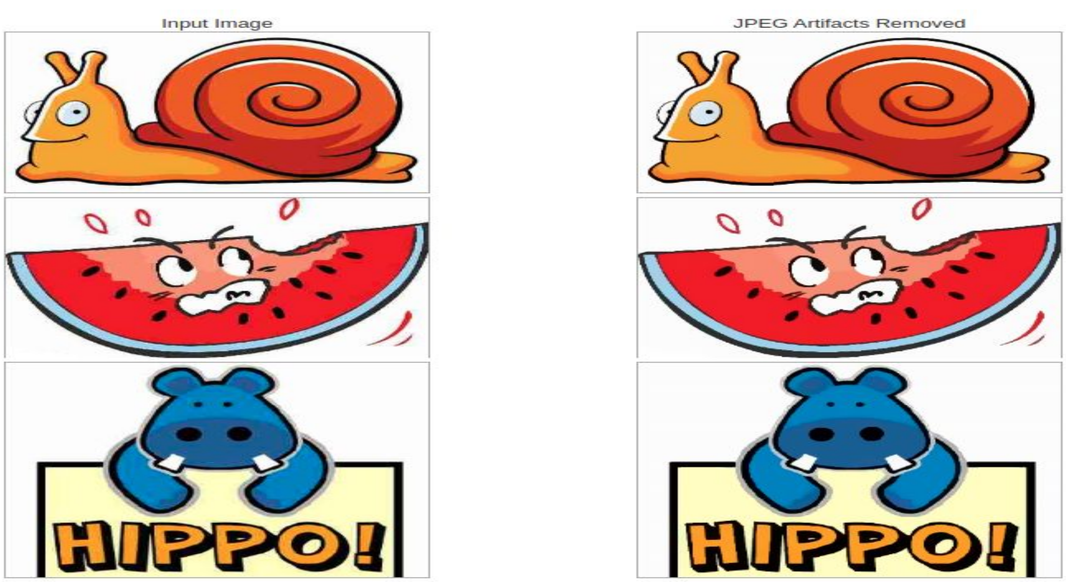
Application in Optical Flow



Application in Stereo Matching



Application in Removing JPEG Artifacts



Work Distribution

- **Gulshan Kumar**
  - Understand and implement a Joint Histogram data structure (initially without Necklace Table) and corresponding algorithms and initial implementation of WMF using only joint histograms.
  - Implement Joint Histogram and BCB using Necklace Table class.
  - Results Table and Experiments
- **Nikhil Bansal**
  - Understand and code Balance count box (without Necklace table) and corresponding algorithms taking into account the joint histogram.
  - Implement class for the Necklace table which will be extended to both Joint Histogram and BCB.
  - Results Table and Experiments

Project correctness and efficiency tests will be performed by both of us at the completion of the project.

Tentative Timeline & Milestones

Week	Milestones
Week 1 (30 Sep - 6 Oct)	• Thoroughly understanding the research paper. • Making models and discussing implementation techniques and plans.
Week 2 (7 Oct- 13Oct)	• Implementing Joint histogram, Balance count box data structures along with required algorithms.
Week 3 (14 Oct - Oct 20)	• Finalization and testing of Joint Histogram and Balance count box • Discuss implementation of the Necklace table (like what to use? Linked lists, arrays, etc.)
Week 4 (21-27 Oct)	Quiz
Week 5 (28 Oct - 4 Nov)	• Start implementing Necklace Tables • Complete Necklace Table class implementation

Week 6 (5 Nov - 9 Nov)	• Change Joint Histograms and BCB to use Necklace Table • Finalize project and bug fixes
Final Week (10 Nov - 15 Nov)	• Test project for efficiency and correctness • Remove Code Smells and Documenting all the things • Prepare Final presentation and Reports

-- End of the Proposal ---