Digital Image Processing
Project Proposal
Team: Mob_Psycho

Project Title: Fast Weighted Median Filtering
Project ID: 12 [Link]

Github Link
https://github.com/Kanav-7/DIP-Project

Team Members
Kanav Gupta (20161151)
Anurag Mehta (20161016)

Main Goal of Project

Main goal of project is to perform weighted median filtering of a given image in an efficient manner. The time complexity of original weighted median filtering (WMF) is O(r^2), where r is size of kernel. In our project we will reduce this complexity to O(r) making it 100+ times faster than general method. For performing this task various Data Structures and mapping will be used which are described later in proposal.

The running time of WMF in our project of this will be shortened from several minutes to less than a second !!

Problem Definition

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, 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 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 following reasons:

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

In this project we will implement few algorithms and data structures to perform WMF. Project will have three major techniques which includes data-weight distribution by a new joint histogram to dynamically allocate weights and pixels, a median tracking algorithm
to reduce time of seeking median by leveraging color coherency of images, and data structure sparsity to reduce the data access cost.

1. **Joint Histogram**

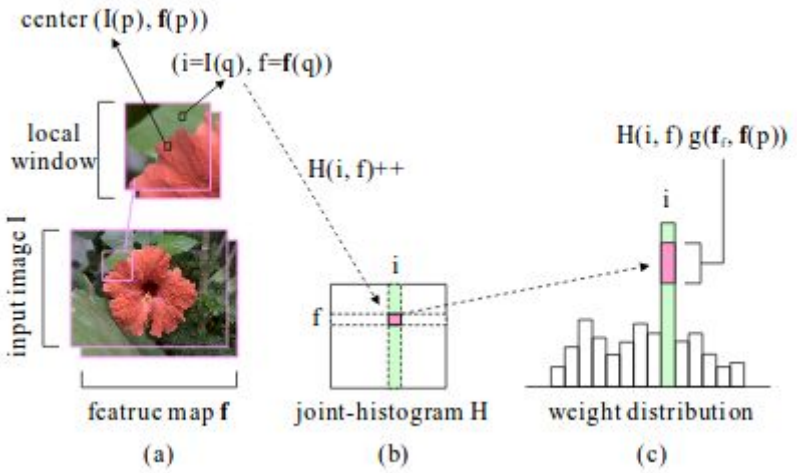
Joint histogram is a 2D histogram structure for storing pixel count. Each row corresponds to feature index and each column corresponds to intensity value of the pixel and each Histogram block stores the count of pixels corresponding to that intensity value and feature.

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

Weights for all i can be obtained using joint histogram as

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

The below diagram clearly explains Joint Histogram



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

2. **Median Tracking**

Second data structure used will be median tracking, Main purpose of this is to find median in efficient way without iterating over joint histogram.

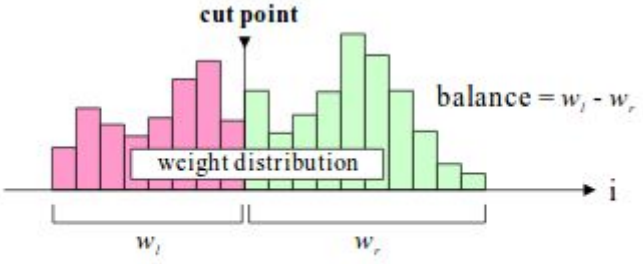
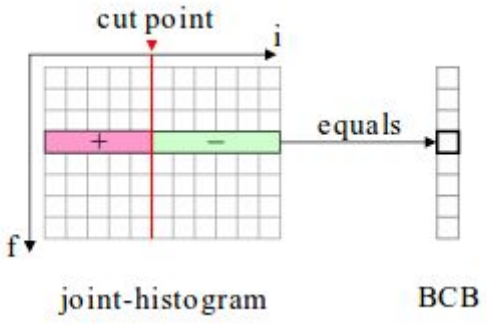


Figure 3. Illustration of the balance and cut point.

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

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



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,

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

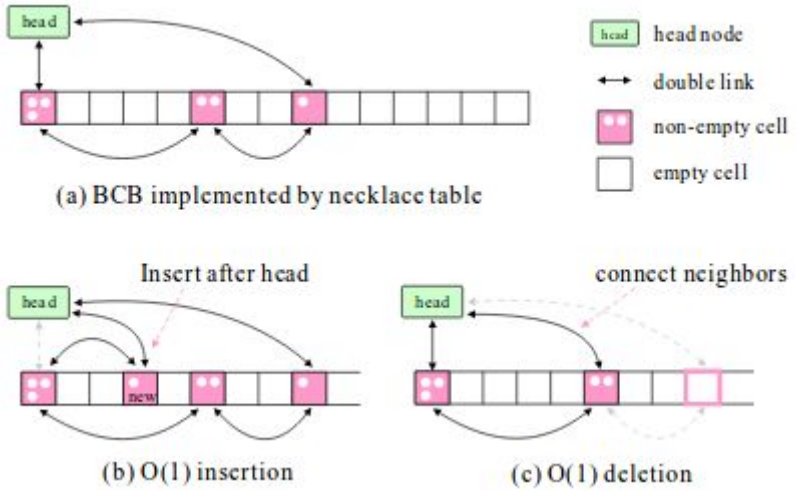
Using BCB balance can be calculated using

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

3. **Necklace Table**

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

Unlike traditional linked-list that keeps an 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.



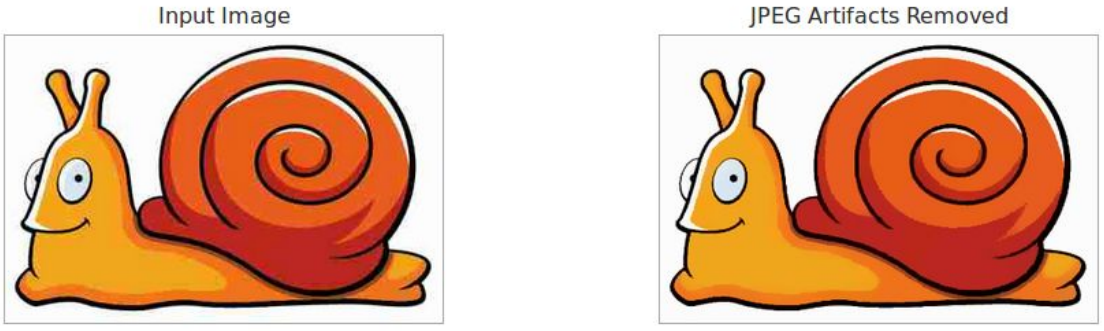
Results of the project

After project completion we will be able to apply weighted median filtering on any data with great efficiency and minimal effect on output. Few Examples from research paper are shown below:

	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%

Efficiency of median tracking and necklace table

Application in Optical Flow



Application in removing JPEG artifacts

Team Members and Tasks for each member

- **Kanav Gupta**

  Understand and implement 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.

- **Anurag Mehta**

  Understand and code Balance count box (without Necklace table) and corresponding algorithms taking in account the joint histogram.

  Implement class for Necklace table which will be extended to both Joint Histogram and BCB.

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

Tentative Timeline

Week 1 (4th October - 10th October)

- Thoroughly understanding the research paper.
- Making model and discussing implementation techniques and plans.

Week 2 (11th October - 17th October)

- Mid Sems

Week 3 (18th October - 24th October)

- Implementing Joint histogram, Balance count box data structures along with required algorithms.

Week 4 (25th October - 31st October)

- Finalization and testing of Joint Histogram and Balance count box
- Discuss implementation of Necklace table (like what to use? Linked lists, arrays etc.)
- Start implementing Necklace Tables

Week 5 (1st November - 7th November)

- Complete Necklace Table class implementation
- Change Joint Histograms and BCB to use Necklace Table

Week 6(8th November- 14th November)

- Finalize project and bug fixes
- Test project for efficiency and correctness

- Make Project Report and Presentations

23rd November or 29th November : Final Presentation
