Fast Weighted Median Filtering

Team: Lumos

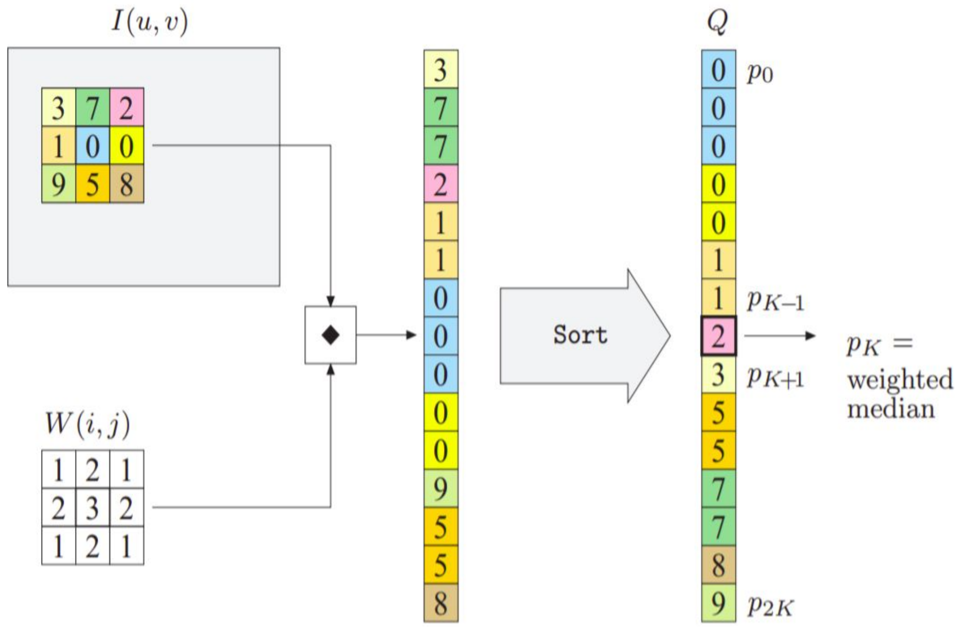
Nikhil Bansal
20161065
CSD

Gulshan Kumar
20161082
CSD

Mentor TA: Aditya Aggarwal

Project Repo https://github.com/gulshan-mittal/Fast-Weighted-Median-Filter

What is Weighted Median Filtering?



Intensities in current window:

{3, 7, 2, 1, 0, 0, 9, 5, 8}

The, weights associated with above intensities

{1, 2, 1, 2, 3, 2, 1, 2, 1}

Sort the intensities.

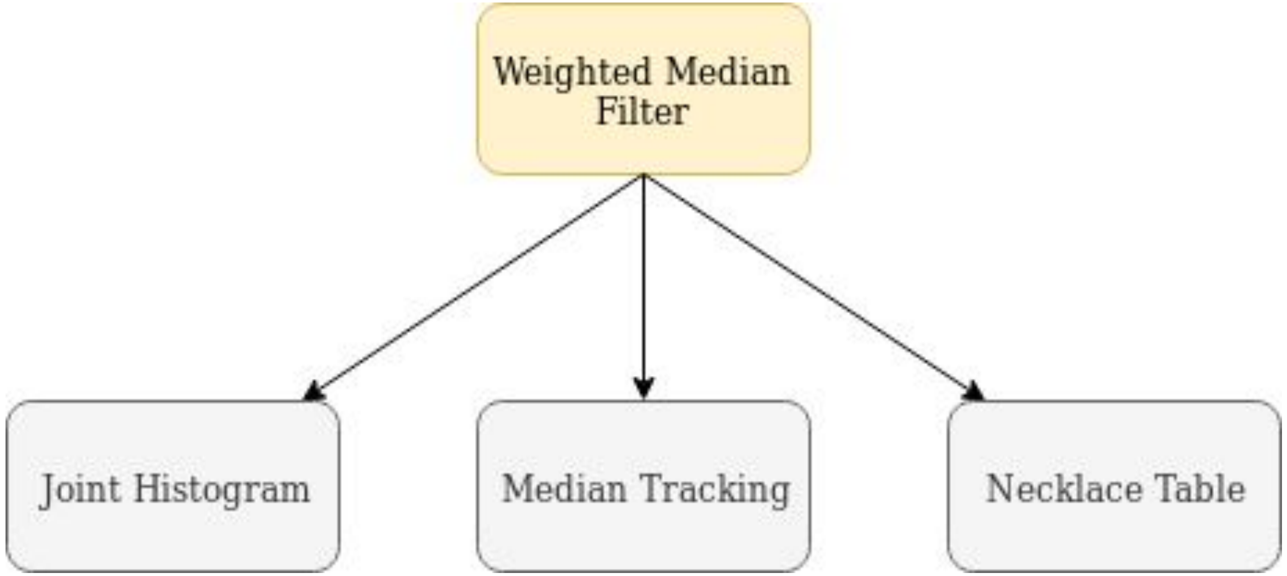
Then, the median is intensity at kth position:

$$\min k \quad \text{s.t.} \quad \sum_{q=1}^{k} w_{pq} \geq \frac{1}{2} \sum_{q=1}^{n} w_{pq}$$

Project Goals

- Re-implement weighted median filtering in efficient manner referencing research paper.
- Reduce time complexity of weighted median filtering (WMF) from O(r*r*log(r)) to O(r), where r is the size of the kernel.
- Showing different applications of Weighted median filtering

Algorithm overview



Improvement Strategies

- Joint Histogram to use box filtering
- Median tracking to cut cumulative histogram procedure
- Necklace table for fast data access in sparse table

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.

- This counting scheme enables fast weight computation even when the window shifts.
- The total weight for all pixels with value index i is :

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

Joint Histogram

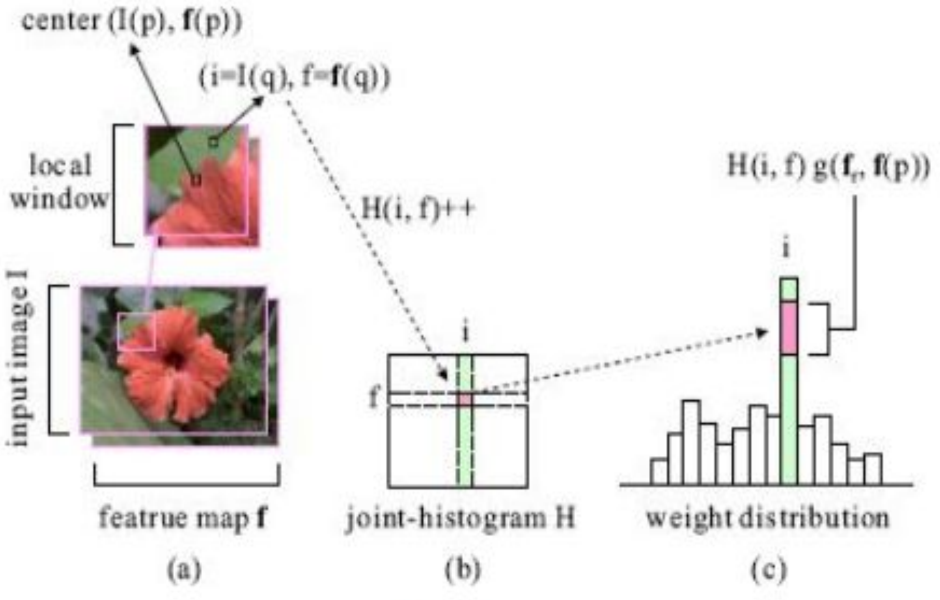


Figure 1.

Algorithms

- Median Finding
- Shift and Update

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

Median Tracking

Median Tracking basically exploits the property that almost all images have similar feature in the adjacent median finding windows to iterate over joint-histogram in a much efficient way.

Cut Point & Balance

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.

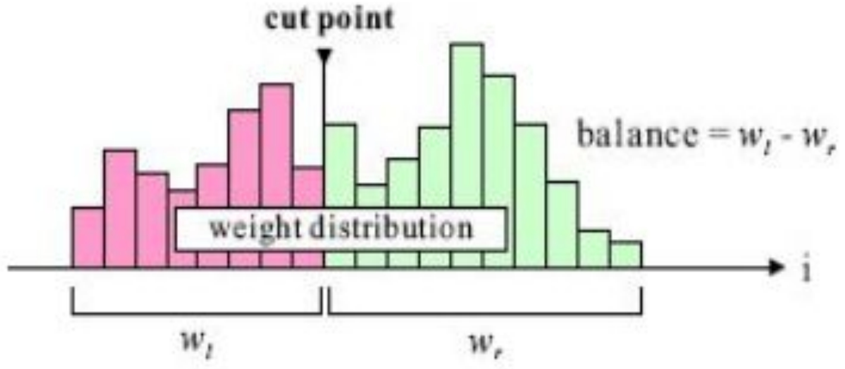


Figure 2. Illustration of the balance and cut point.

Balance Computation Box (BCB)

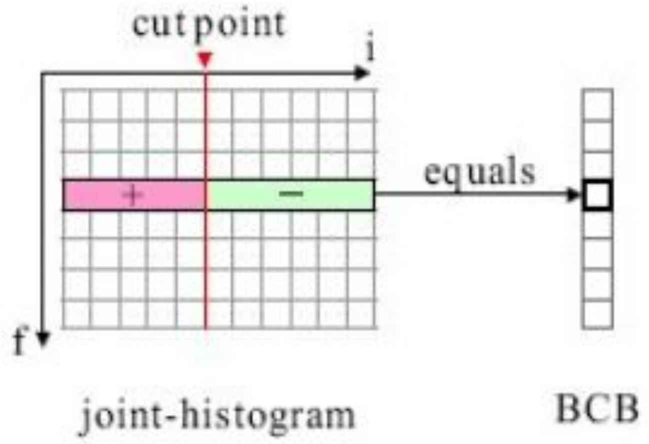


Figure 3. Relationship between BCB and our joint-histogram. By separating each row into the left and right parts according to the cut point in the joint-histogram, their total count difference is put into the corresponding cell in BCB.

Balance Computation Box (BCB)

Stores the difference of pixel numbers on the two sides of the cut point in the corresponding row of the joint-histogram.

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

Using BCB balance can be calculated as:

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

where $g(f_f, f(p))$ is a weight map.

Necklace Table

- Exploit the data sparsity in joint histogram and BCB
- Reduces the required space as well as time required to access a particular value.
- O(1) data access
- O(1) element insertion
- O(1) element deletion

Necklace Table

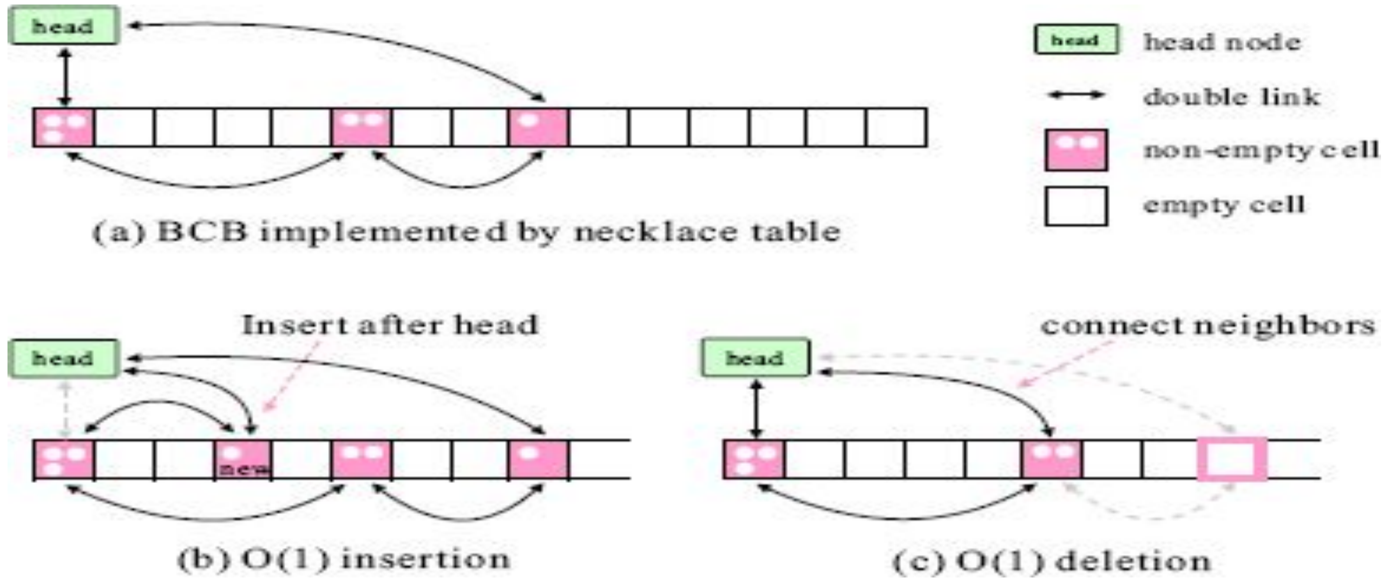
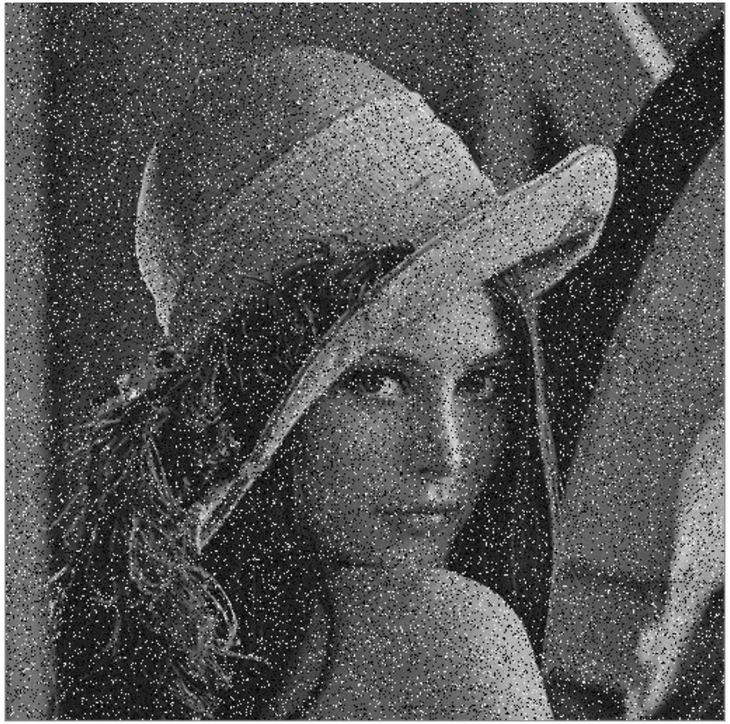
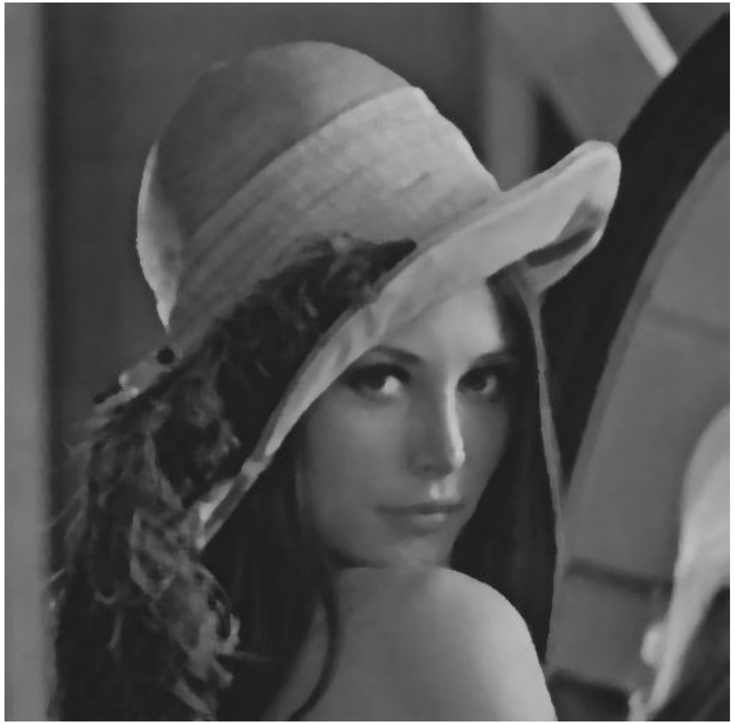


Figure 4. Demonstration of the necklace table.

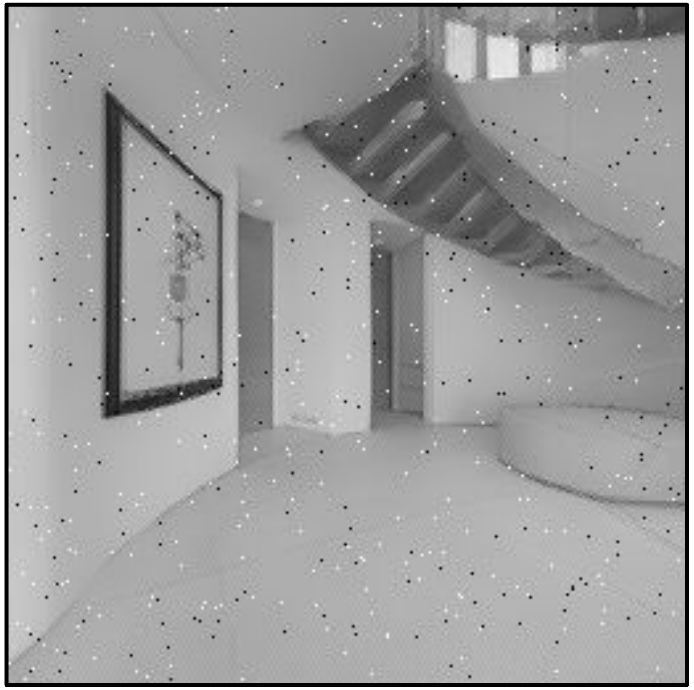
Results Obtained

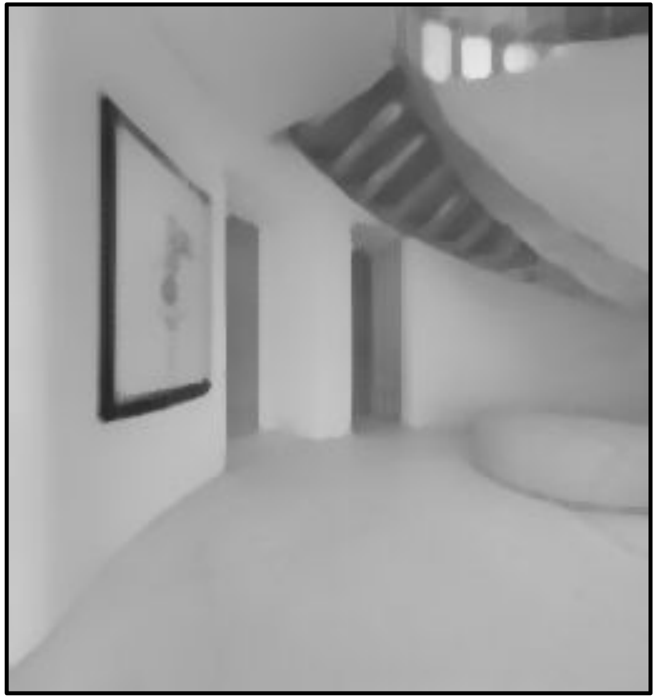
Salt & Pepper Denoising



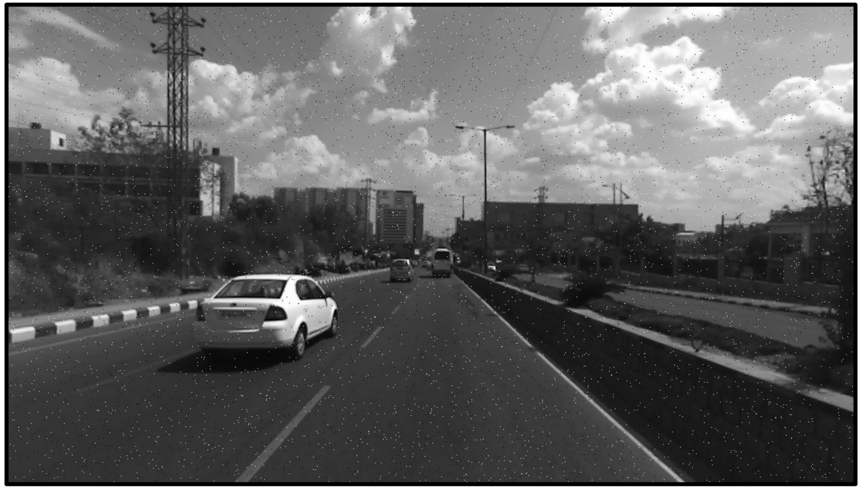


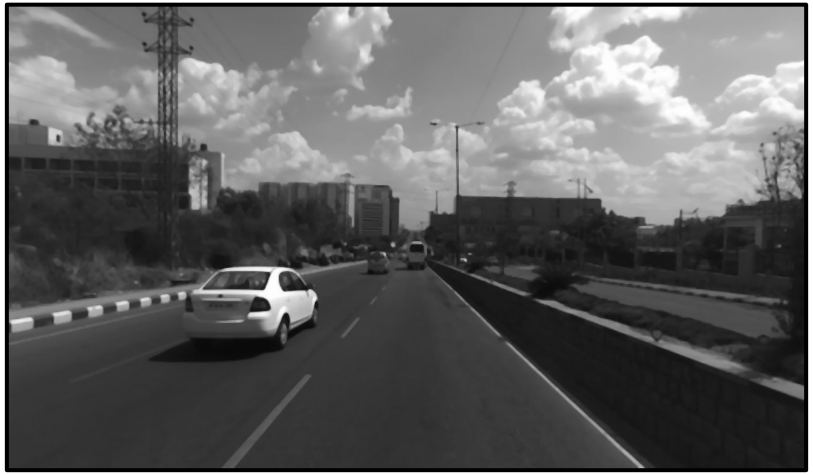
Salt & Pepper Denoising (Gibson dataset)



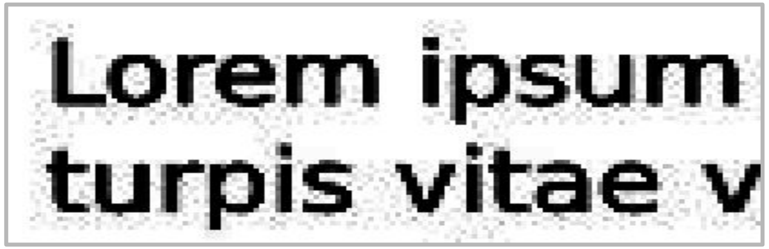


Salt & Pepper Denoising (Indian dataset)

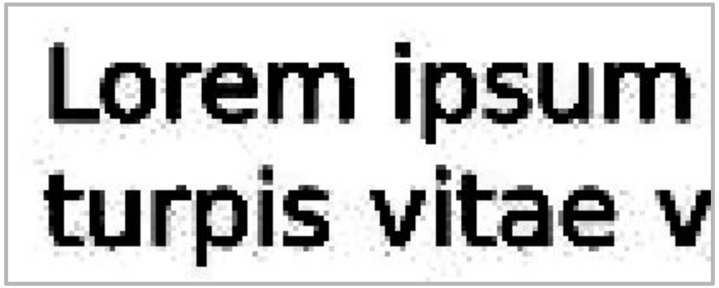




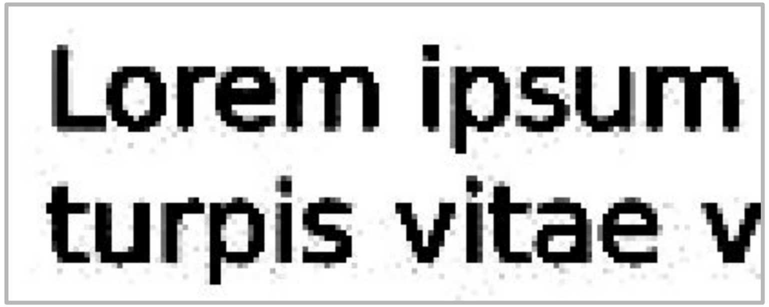
JPEG Artifact Removal



Lorem ipsum
turpis vitae v



Lorem ipsum
turpis vitae v

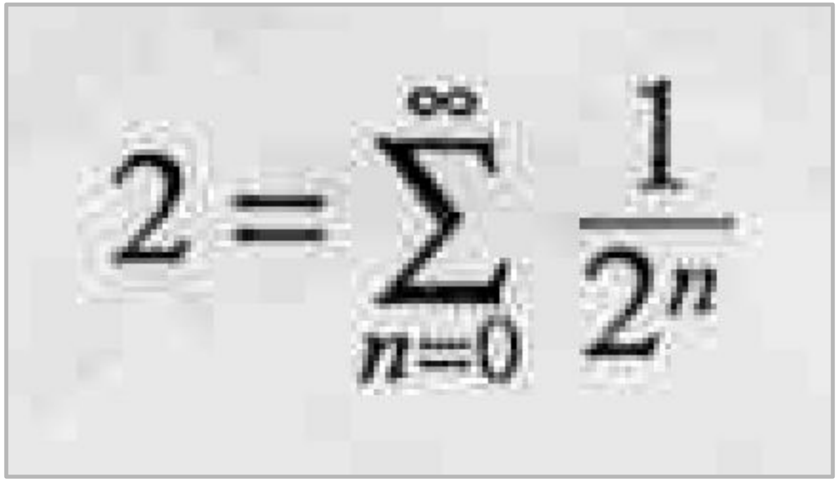


Lorem ipsum
turpis vitae v

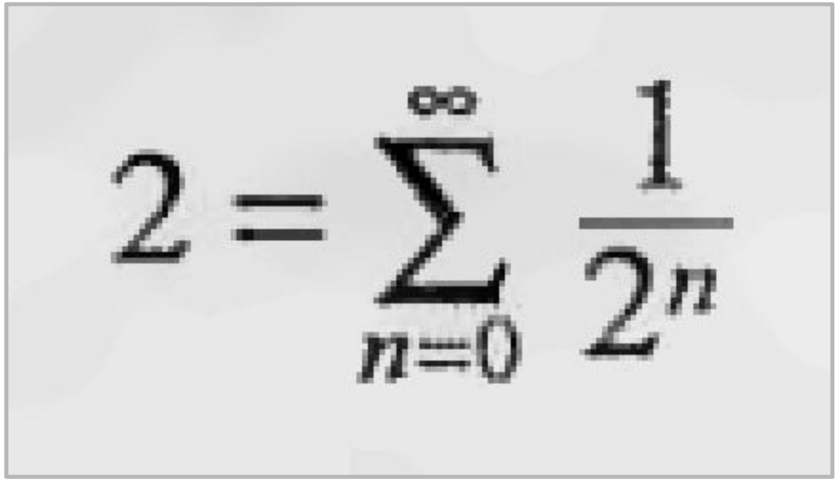
Gaussian Kernel with Radius 10

Gaussian Kernel with Radius 32

JPEG Artifact Removal



$$2 = \sum_{n=0}^{\infty} \frac{1}{2^n}$$

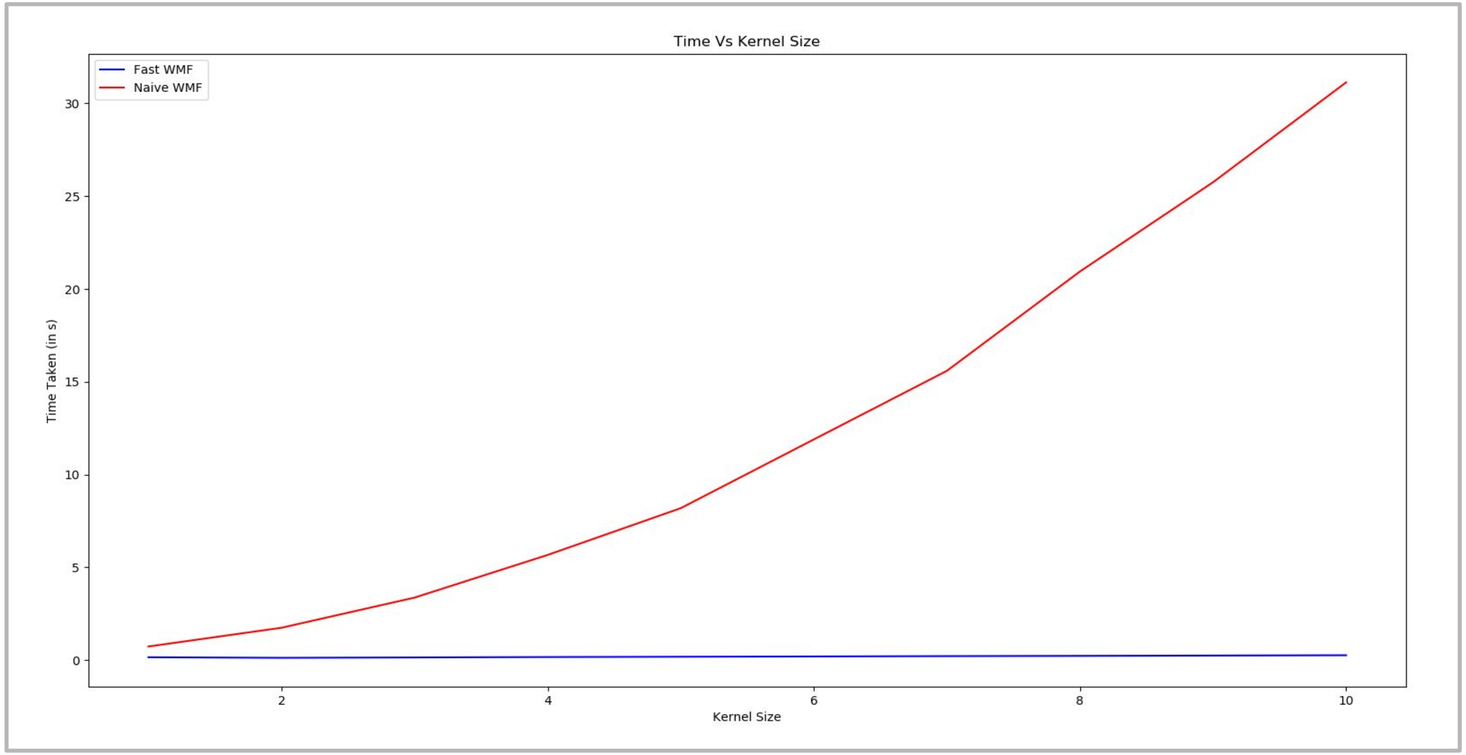


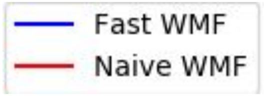
$$2 = \sum_{n=0}^{\infty} \frac{1}{2^n}$$

Gaussian Kernel with Radius 20

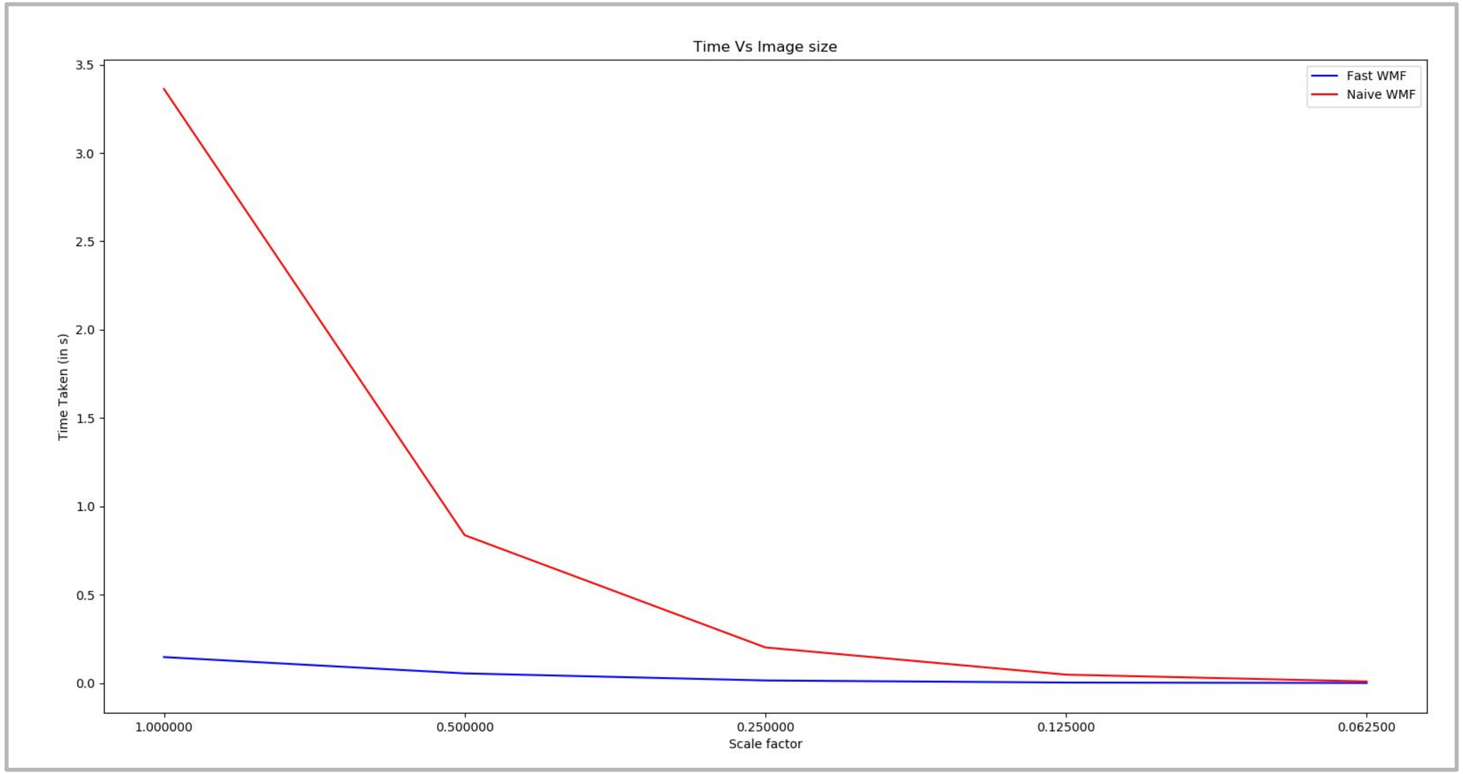
Analysis

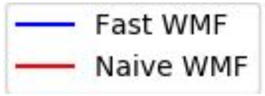
Time Vs kernel Size



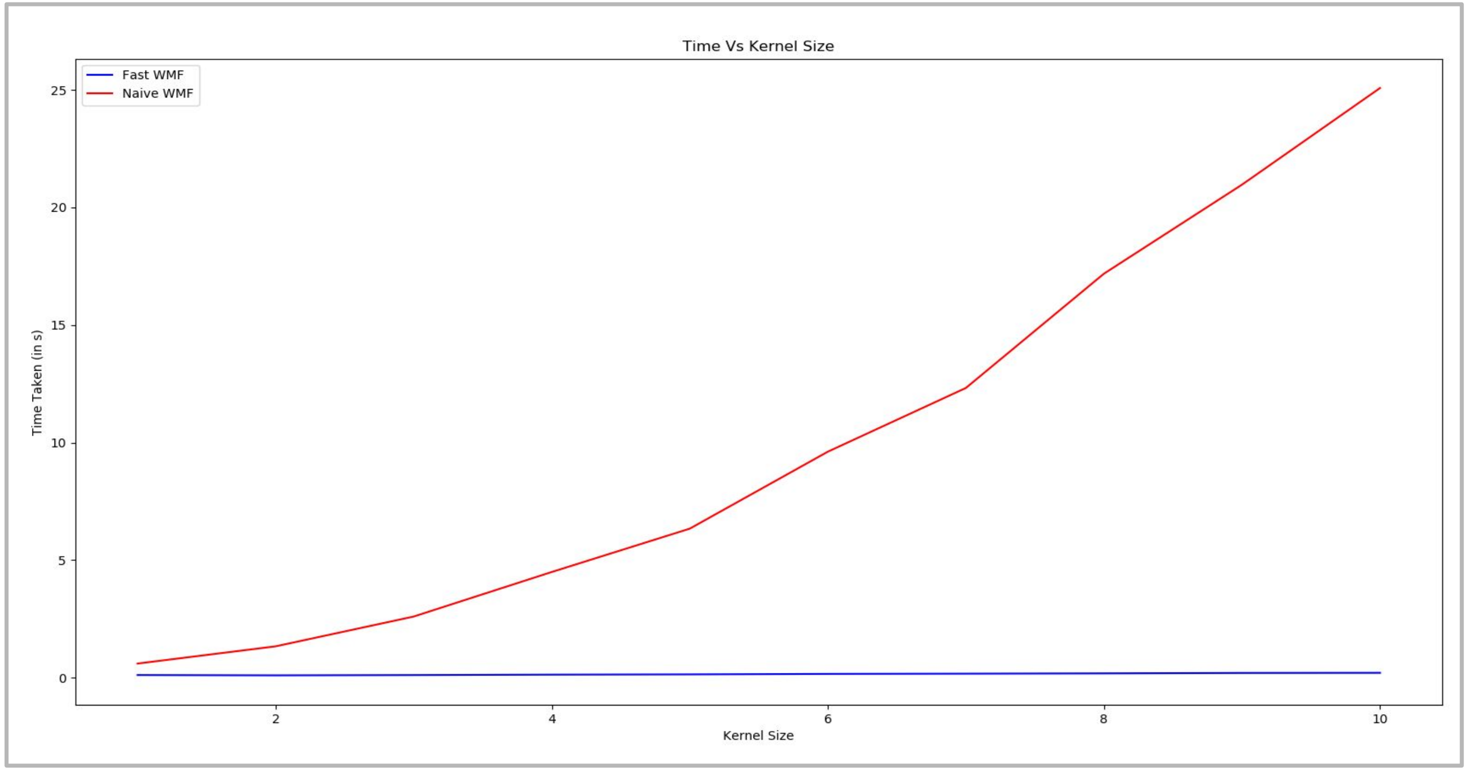


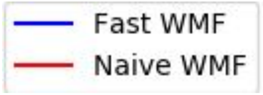
Time Vs Image Size (Scale)



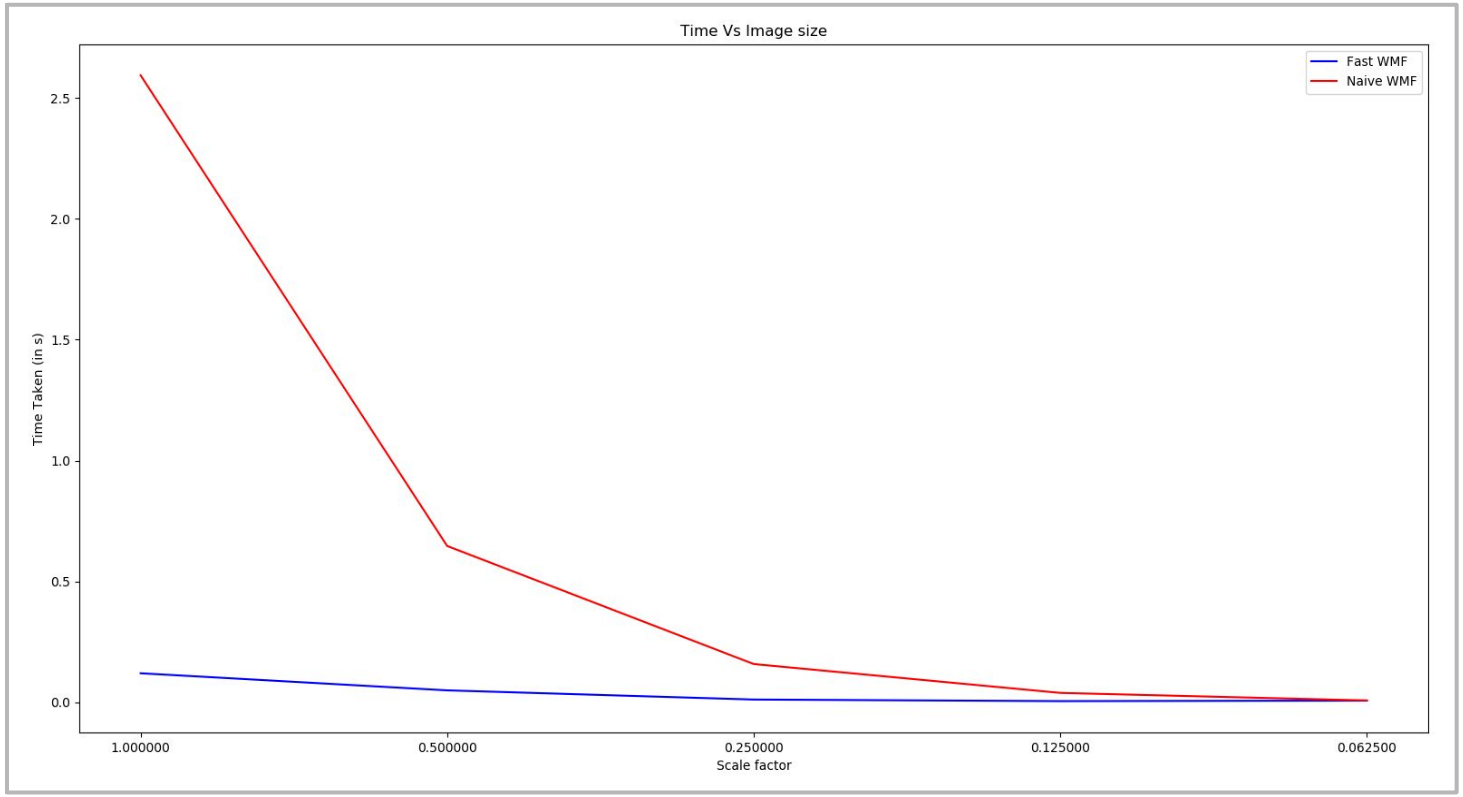


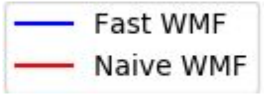
Time Vs kernel Size (Gaussian Kernel)





Time Vs Image Size (Gaussian Kernel)





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, Analysis & plots

- 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.
    - Baseline: Naive Implementation