Fast Weighted Median Filtering

Team Mob_Psycho

- Kanav Gupta
- Anurag Mehta

TA Mentor: Abhijeet Kumar

Introduction

What is Weighted Median Filtering?

Consider intensities in current window centered at some p as
{ 10 ; 26 ; 122 ; 234 ; 256 ; }
with each intensity having weights as
{ 0.15 ; 0.1 ; 0.2 ; 0.3 ; 0.25 ; }

So median is intensity at point p* where

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

In the above example median intensity will be 234 corresponding to weight 0.3

Intro

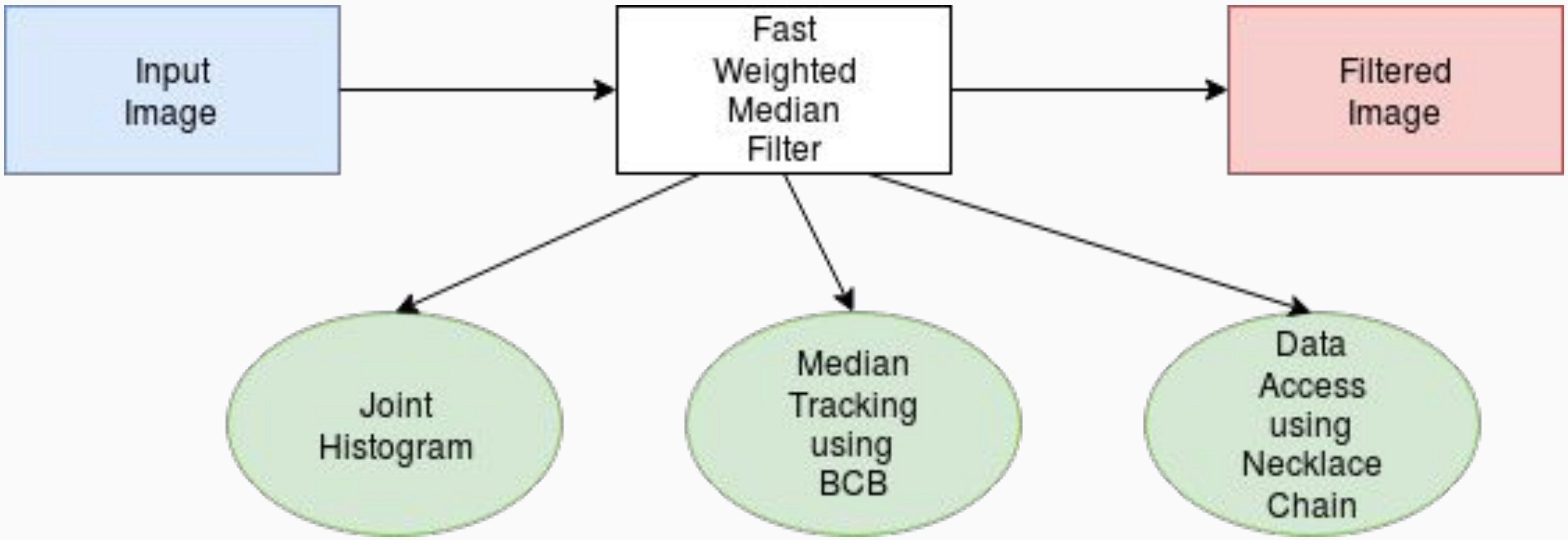
In our project we have re-implemented Weighted Median FIlter for images in a much efficient way with reference to the research paper. In our project we have reduced time complexity of WMF from O(r*r*logr) to O(r) which in most general cases 100+ times faster.

Motivation

- Modern World of Digital Photography
- Transmission technology prone to noise
- Use in various computer vision problems like stereo matching, optical flow estimation, image denoising etc.
- Method used for making unweighted median filtering/mean filtering cannot be applied here

Approach Used

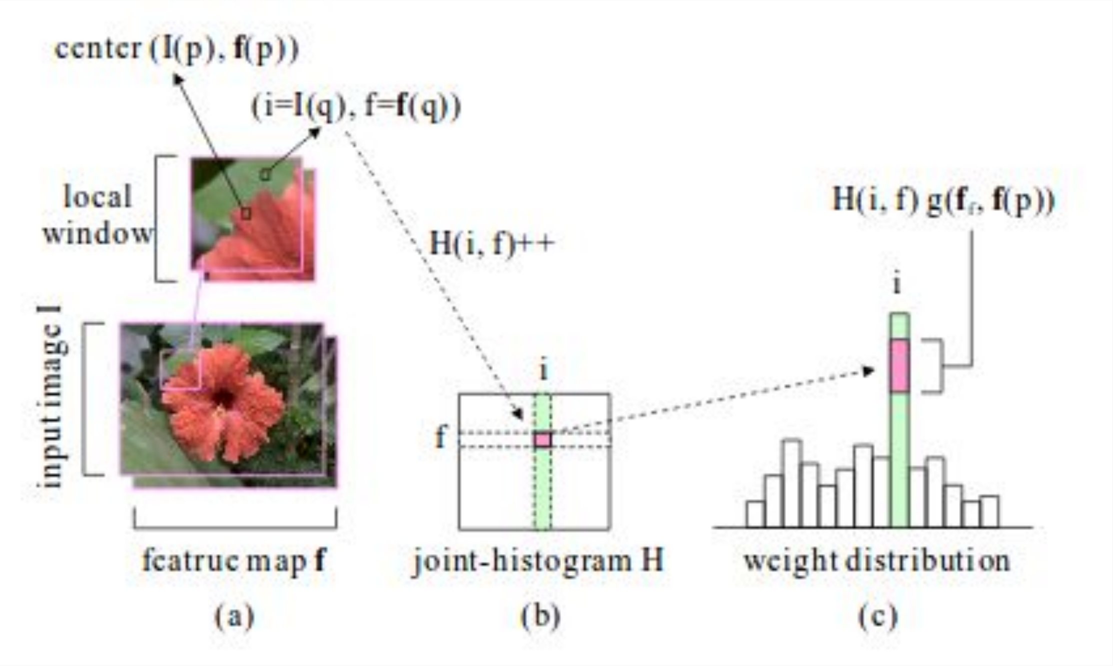
Overview


Input Image
Fast Weighted Median Filter
Filtered Image
Joint Histogram
Median Tracking using BCB
Data Access using Necklace Chain

Joint Histogram

- Joint histogram is a two dimensional which stores the pixel count in its bins according to the corresponding features
- Each row of histogram corresponds to feature index and each column corresponds to intensity value of the pixel.

Diagram



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

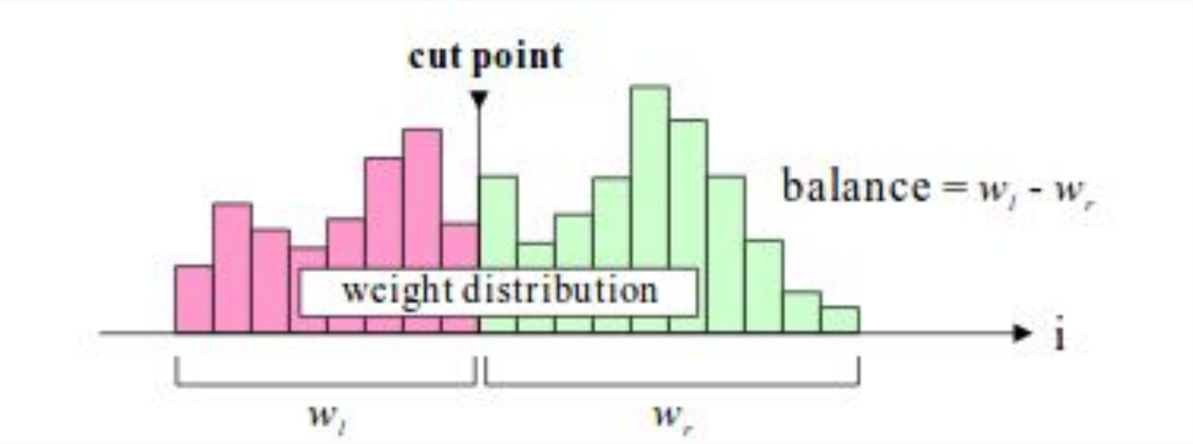
Algorithms

- Median Finding
- Shift and Update

Median Tracking

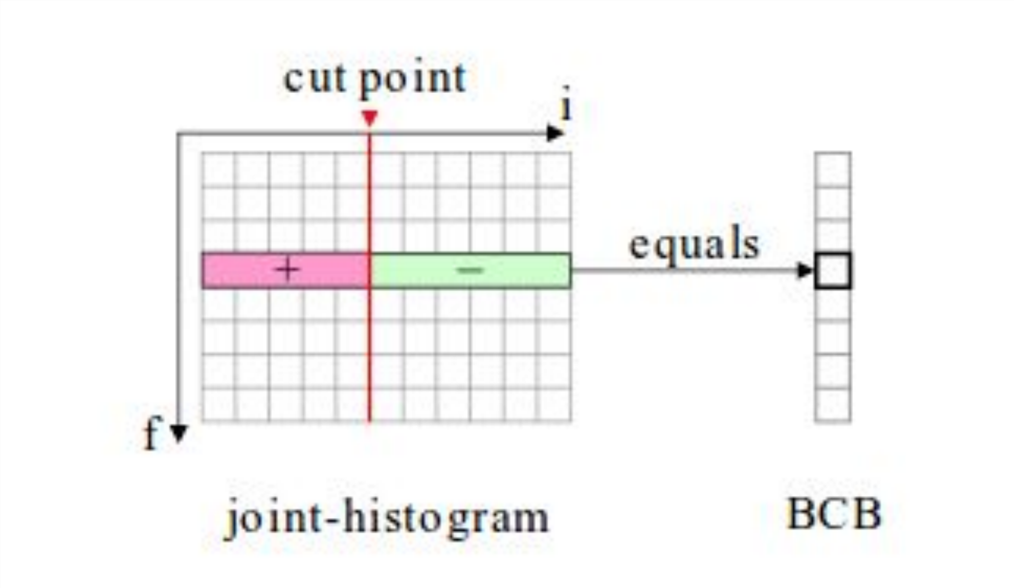
Median tracking basically exploits the property that almost all images have similar features in the adjacent median finding windows to iterate over joint-histogram in a much efficient way. Median tracking work on concept of Cut point and balance.

Cut Point and Balance



- Tracking Median
- Efficient calculation of Balance

Balance Counting Box



Balance Counting Box

Balance Counting Box stores difference of pixels on both side cut points

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

So balance can be calculate simply as

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

Where g is weight map

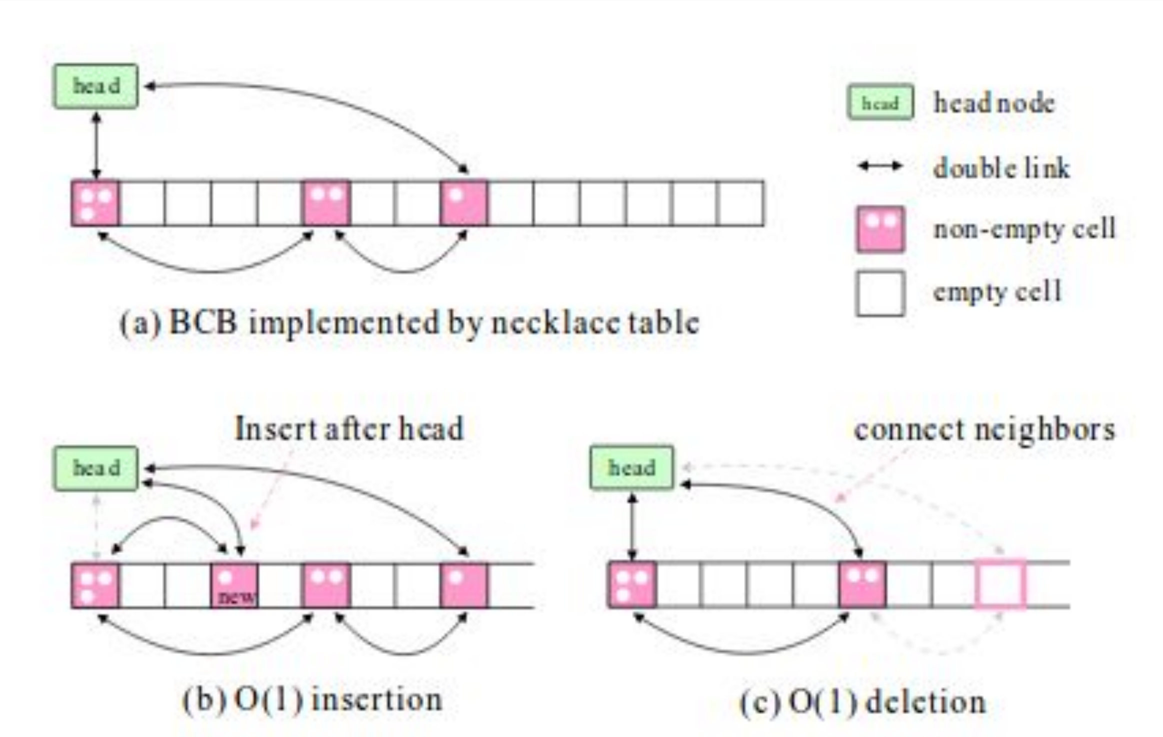
Necklace Table

Exploit the data sparsity in joint histogram and BCB and reduces the required space as well as time required to access a particular value.

Features

- O(1) data access
- O(1) element insertion
- O(1) element deletion
- O(N) traversal

Insertion/Deletion

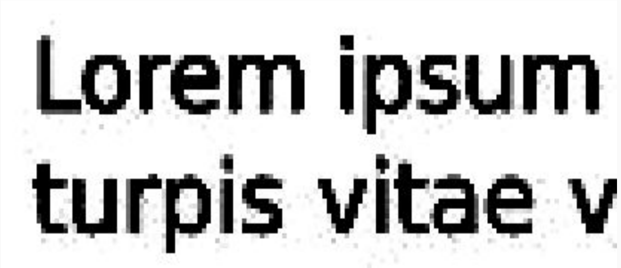


Results Obtained

Salt and Pepper denoising

JPEG artifacts removal

Gaussian Filter with radius = 5



Gaussian Filter with radius = 25

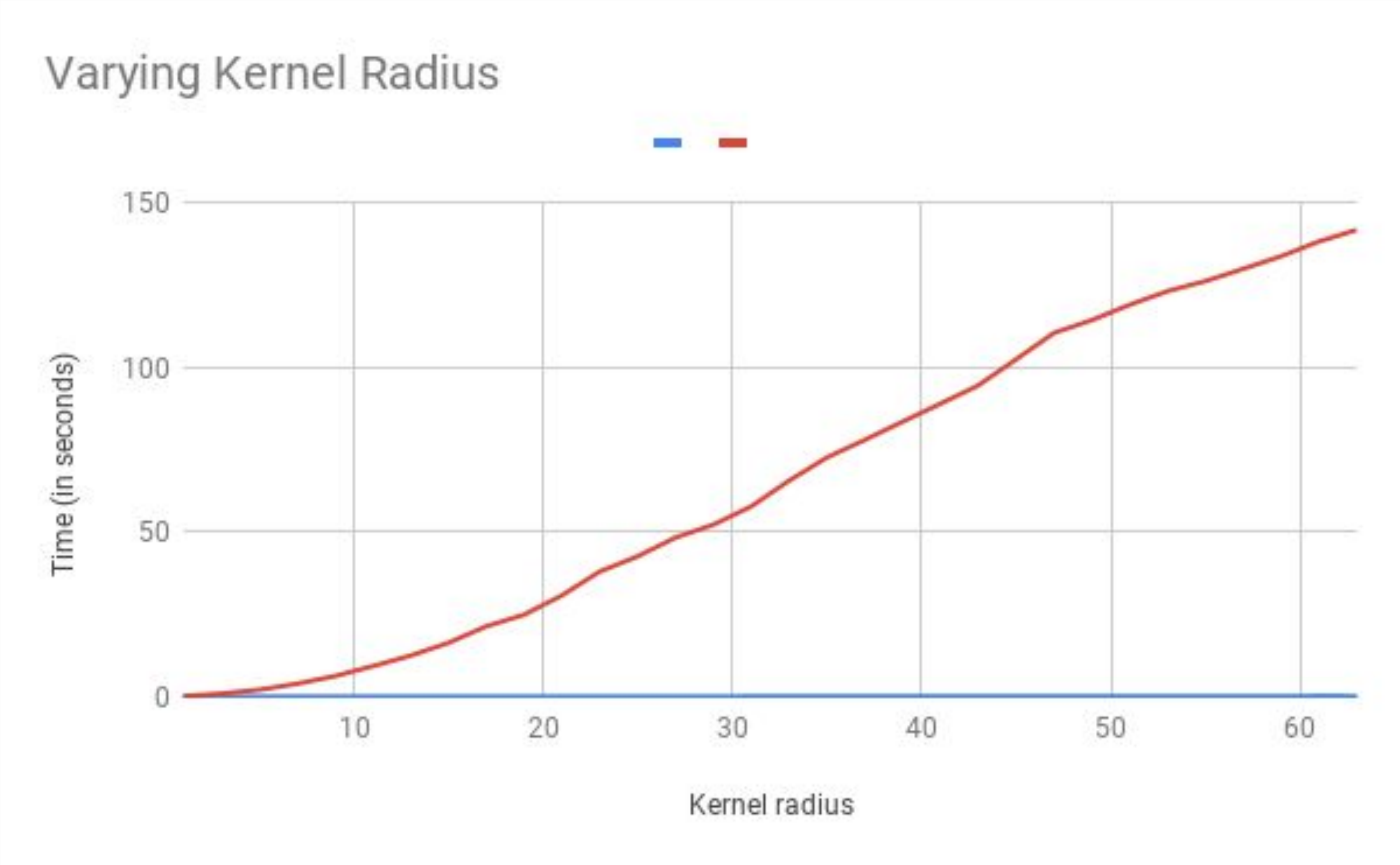
JPEG artifacts removal

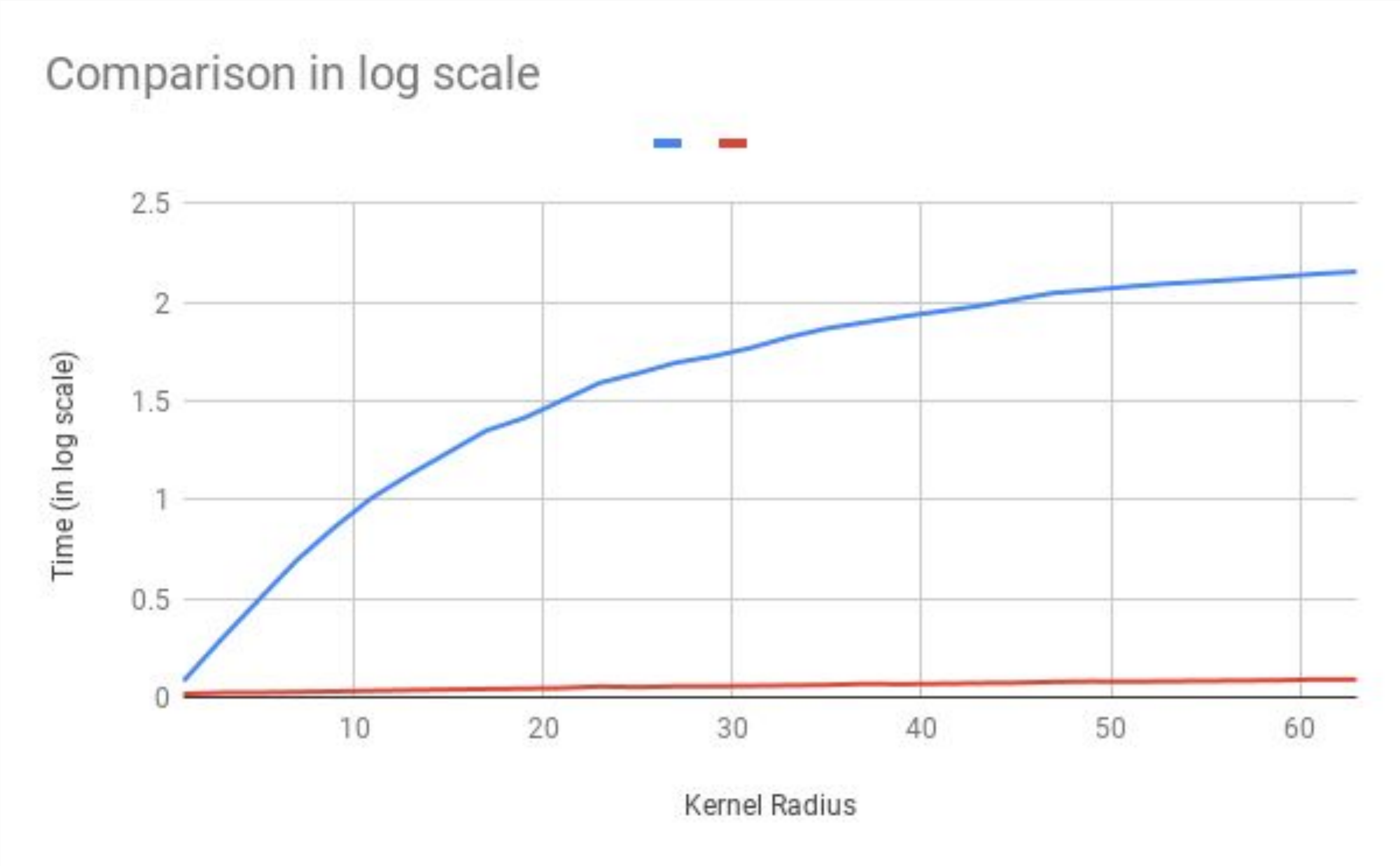
$$2 = \sum_{n=0}^{\infty} \frac{1}{2^n}$$

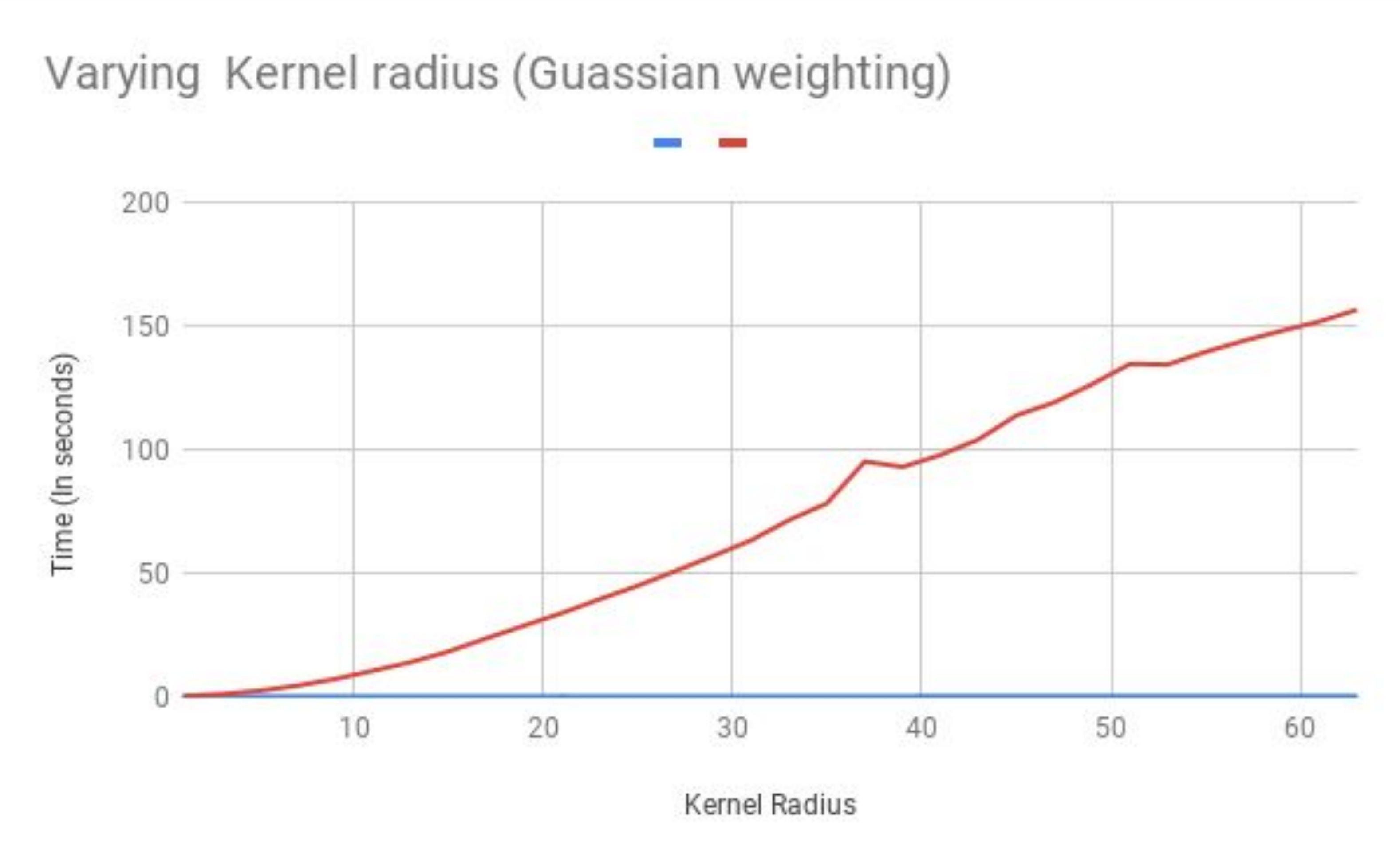
$$2 = \sum_{n=0}^{\infty} \frac{1}{2^n}$$

Gaussian Filter with radius = 25

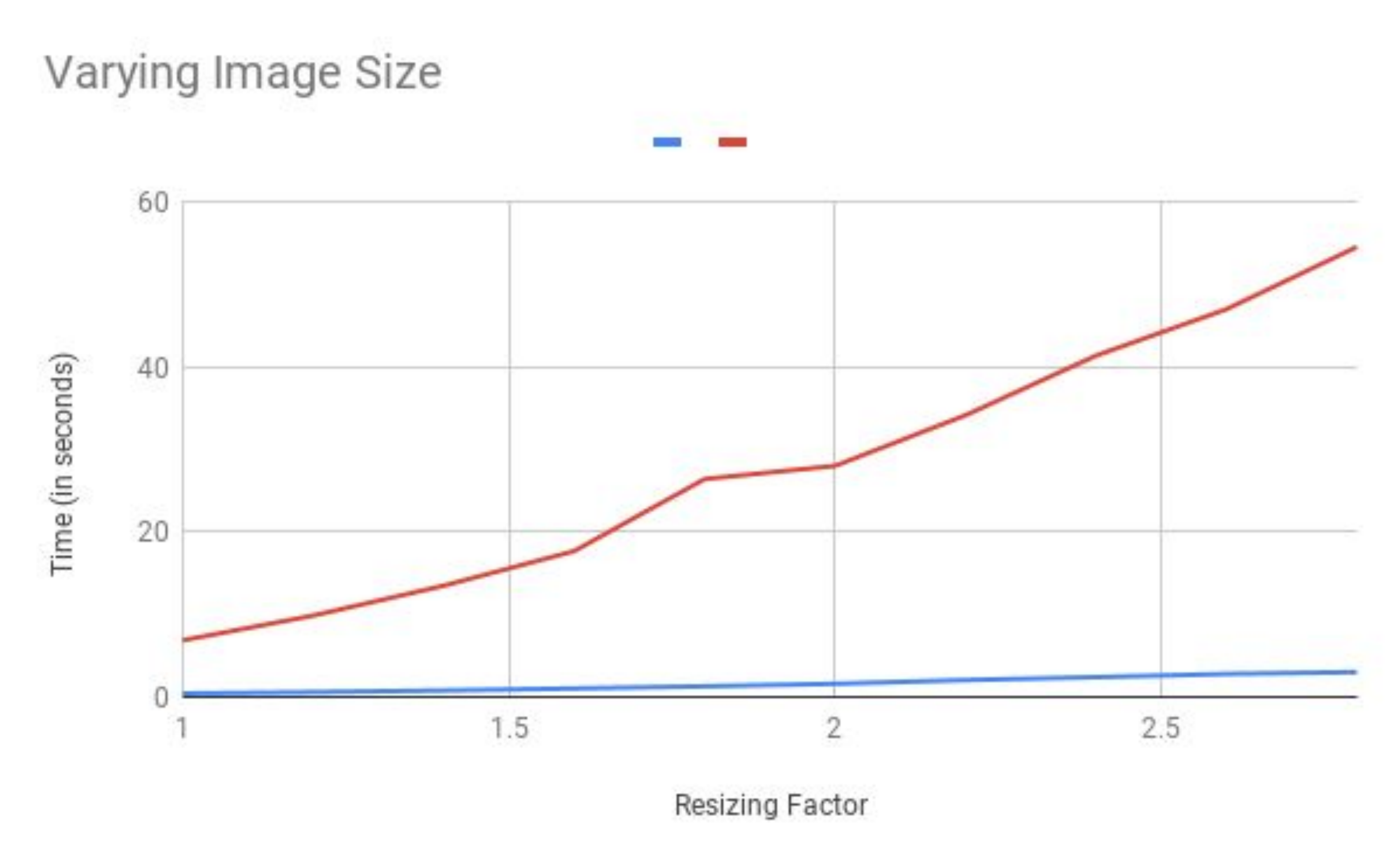
Experiments


Varying Kernel Radius
150
100
50
0
Time (in seconds)
10
20
30
40
50
60
Kernel radius


Comparison in log scale
2.5
2
1.5
1
0.5
0
10
20
30
40
50
60
Time (in log scale)
Kernel Radius



Varying Kernel radius (Guassian weighting)
Time (In seconds)
200
150
100
50
0
10
20
30
40
50
60
Kernel Radius



Varying Image Size
60
40
20
0
Time (in seconds)
1
1.5
2
2.5
Resizing Factor


Varying Image size(Gaussian Kernel)
100
75
50
25
0
time taken (in seconds)
2
2.5
3
3.5
4
4.5
scaling factor

Thanks :)