Optimization

Edge Filter

W x H	speed up
128 × 128	2.07
256 × 256	2.89
512 × 512	2.56
768 × 768	2.36
1024 × 1024	2.57

Guassian Filter

W x H	speed up
128 × 128	2.40
256 × 256	2.49
512 × 512	2.70
768 × 768	2.51
1024 × 1024	2.62

BoxBlur Filter

W x H	speed up
128 × 128	2.45
256 × 256	2.45
512 × 512	2.46
768 × 768	2.70
1024 × 1024	2.60

Sharpen Filter

W x H	speed up
128 × 128	2.39
256 × 256	2.66
512 × 512	2.51
768 × 768	2.76
1024 × 1024	2.58

Identify Filter

W x H	speed up
128 × 128	2.45
256 × 256	2.47
512 × 512	2.51
768 × 768	2.71
1024 × 1024	2.89

Optimize Approach

Using Cache: Previously, the outer loop was x and the inner loop was y. It was changed to have the outer loop as y and the inner loop as x, preserving spatial locality.

Achieved a speedup of up to 1.6 with caching alone.

Using Loop Unrolling: In multiple for-loops, increased the inner loop's step size and repeated the instructions inside to reduce branch instruction overhead.

Achieved a speedup of up to 1.8 with loop unrolling alone.

Original code: Had branches proportional to width * height * 3 ∗ 3.

Optimized code: Reduced branches to width / 2 * height.

Branch overhead decreased by a factor of 18 (9 ∗ 2).

If the code length were extended significantly to accommodate width / 32 * height branches, Branch overhead would decrease by a factor of 288 (9 * 32), but this was unnecessary due to readability concerns (image W, H are multiples of 32).

Using Cache & Loop Unrolling:

Achieved a speedup of up to 2.1 when both techniques were applied simultaneously.

Using Cache & Loop Unrolling with Cache: Ensured optimization with spatial locality during loop unrolling as well.

Applying these techniques achieved a speedup of up to 2.8.