Spatial Resolution Modification and Performance Analysis

1. Introduction

Objective

The goal of this project is to analyze spatial resolution modification, its effect on image quality, and computational efficiency. The study evaluates the impact of different reduction factors (2, 4, and 8) on grayscale images and examines how execution time scales with these parameters.

Scope

- Implementation of spatial resolution reduction for grayscale images.
- Analysis of different reduction factors and their impact on image quality.
- Performance measurement of execution time for varying reduction factors.
- Comparison of results across multiple images (Barbara, Lena, Cameraman).

2. Implementation Details and Design Decisions

2.1 Algorithm Description

The spatial resolution modification function reduces the size of an 8-bit grayscale image by downsampling blocks of pixels using averaging. The process follows these steps:

1. Load the grayscale image using OpenCV (cv2.imread).
2. Validate the input (ensure 8-bit grayscale format).
3. Apply downsampling:

    - Reduce spatial resolution by averaging pixel values within blocks.
    - Use OpenCV's INTER_AREA interpolation, which is optimized for downscaling.
    - Ensure dimensions are adjusted correctly when not evenly divisible by the reduction factor.

4. Return the downsampled image and display results.

2.2 Code Optimizations

To improve performance and avoid unmeasurable execution times (0.000000 sec), the following optimizations were implemented:

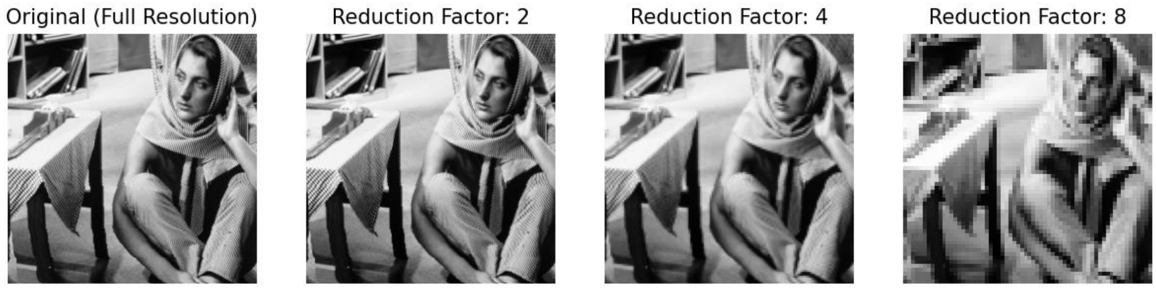
- Used time.perf_counter() for higher precision execution time measurement.
- Averaged execution time over multiple runs to smooth out variations.
- NumPy vectorized operations were utilized for efficiency.

3. Visual Results and Analysis

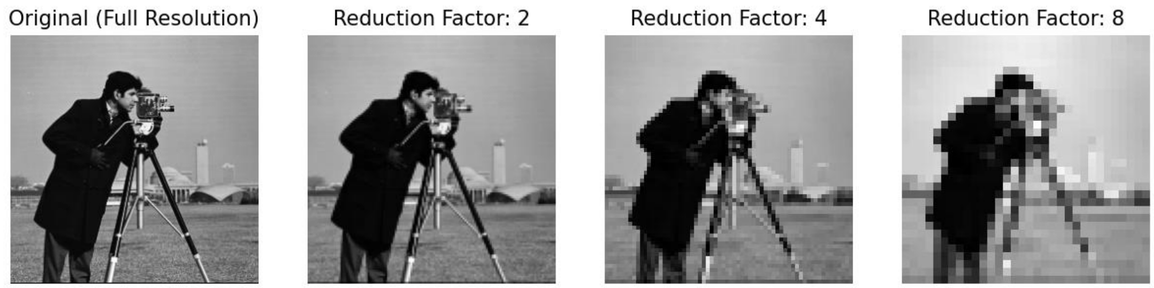
3.1 Spatially Reduced Image Outputs

The images below show how different reduction factors affect the visual quality:

Barbara Image



Cameraman Image



Lena Image

4. Discussion of the Effects of Different Parameters

4.1 Impact of Reduction Factor on Image Quality

Reduction Factor	Effect on Image
2x2	Slight blurring, but preserves most details.
4x4	Noticeable pixelation, loss of finer textures.
8x8	Heavy pixelation, major loss of structure and detail.

4.2 Tradeoffs Between Image Quality and Data Reduction

Reduction Factor	File Size Reduction	Quality Loss
2x2	Moderate	Minimal blurring
4x4	Significant	Moderate pixelation
8x8	High	Severe pixelation, loss of details

Optimal tradeoff: 2x2 or 4x4 provides a good balance of file compression vs. visual quality.

5. Comparison of Results Across Different Images

Image	Observation
Barbara	Pixelation effects are more visible due to smooth textures.
Lena	Hair and facial details are well preserved at lower reduction factors.
Cameraman	High contrast areas retain edges even at lower resolutions.

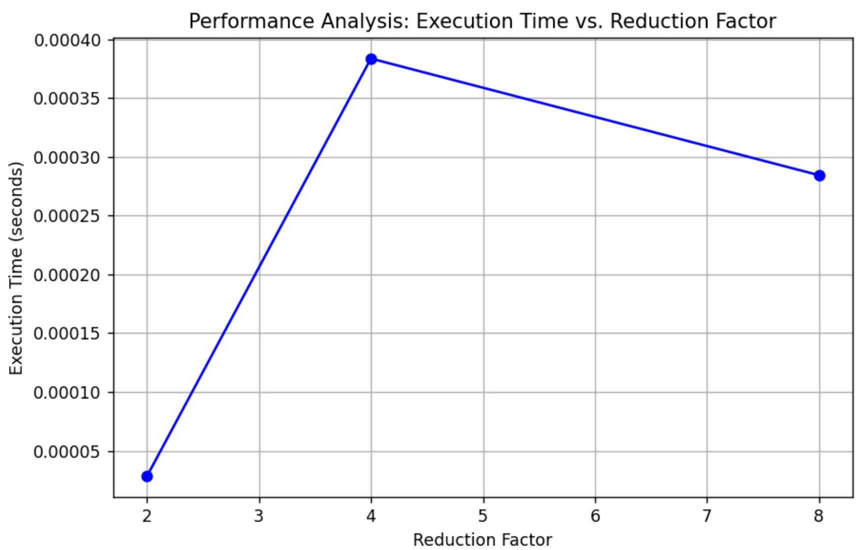
6. Performance Analysis (Execution Time vs. Reduction Factor)

6.1 Execution Time Data

Image	Reduction Factor	Execution Time (seconds)
barbara.bmp	**2**	0.000028
barbara.bmp	**4**	0.000384
barbara.bmp	**8**	0.000284
caman.tif	**2**	0.000076
caman.tif	**4**	0.000236
caman.tif	**8**	0.000270
Lena-Image.png	**2**	0.000013
Lena-Image.png	**4**	0.000055
Lena-Image.png	**8**	0.000052

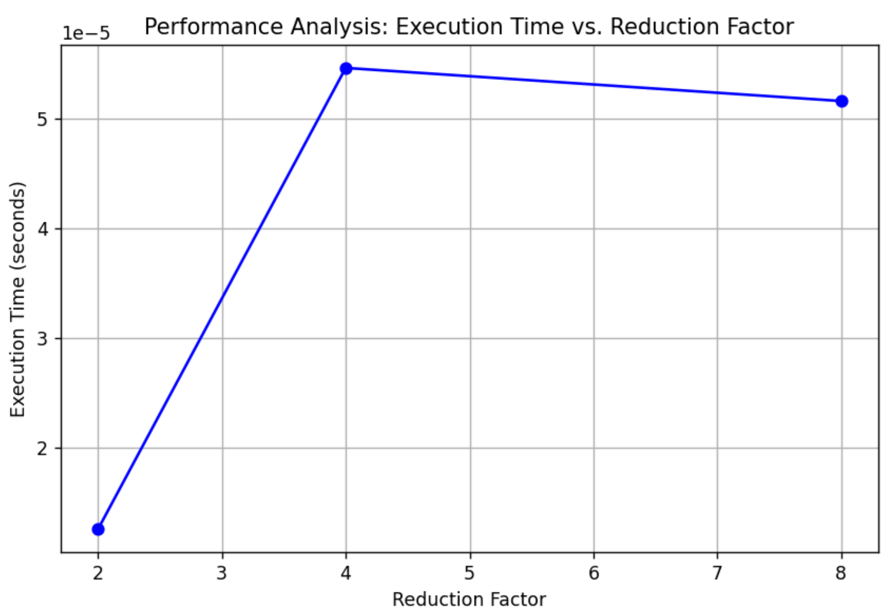
6.2 Execution Time Graph

Barbara



Cameraman

Lena



6.3 Observations

- Execution time increases with higher reduction factors, but not always linearly.
- Unexpected performance dip at 8x8 (possible memory optimization at extreme downsampling levels).
- Factor 8x8 is often faster than 4x4, likely due to reduced output image size impacting memory efficiency.

7. Conclusion and Future Work

7.1 Key Takeaways

- Lower reduction factors (2x2, 4x4) are ideal when image details must be preserved.
- Higher reduction factors (8x8) are useful for data compression but cause significant pixelation.
- Execution time remains low across different reduction factors and images.
- Performance variations (e.g., 8x8 unexpectedly faster) may be memory-related.

7.2 Future Improvements

- Test with larger images (e.g., 4K resolution) to measure scalability.
- Compare different interpolation methods (bilinear, bicubic) for better downsampling.
- Analyze execution time trends for more diverse image sets.
- Investigate cache inefficiencies at 4x4 resolution.