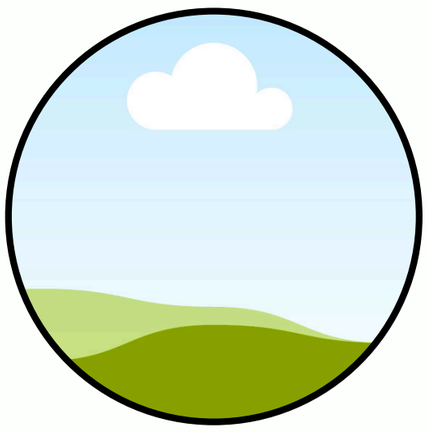
IMAGE SHARPENING IN RUST

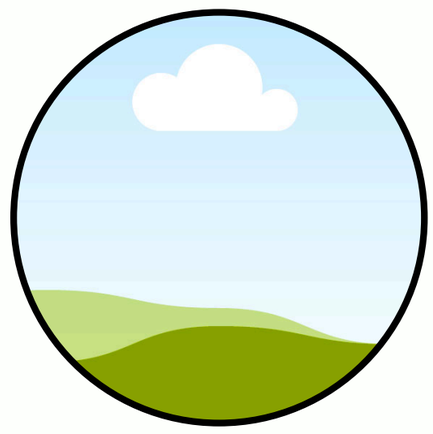


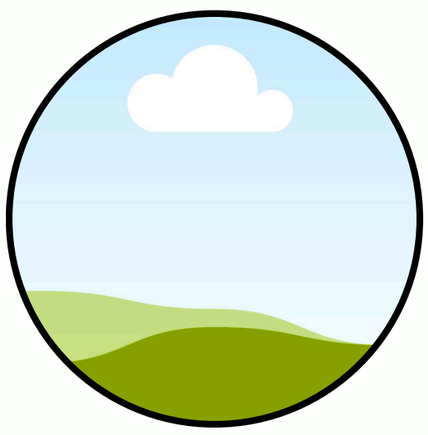
WHAT THE WORK IS ABOUT

This part focuses on implementing image sharpening in Rust. Image sharpening enhances the edges and details of an image. The goal is to explore parallel and sequential processing techniques

HOW IT IS ACCOMPLISHED

First I am using get_image function to read and decode the image. Next, I am using the non parallel sharpening function to sharpen the image sequentially.



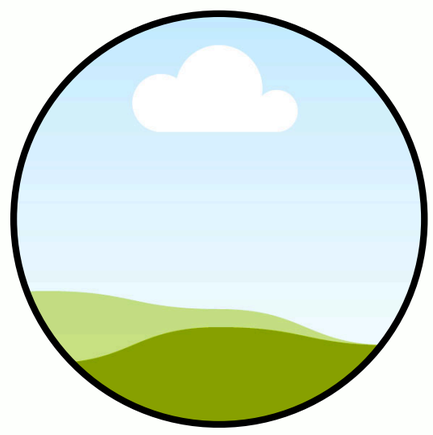


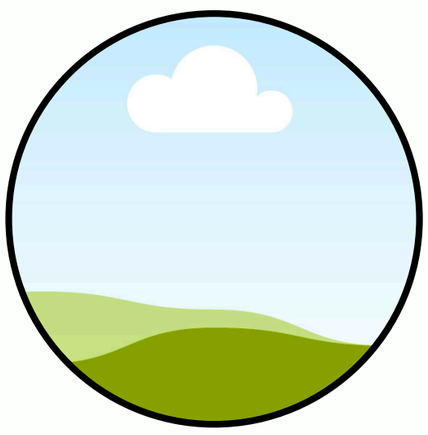
HOW IT IS ACCOMPLISHED

Next, I am using the parallel sharpening function to sharpen the image in parallel Finally, measuring the run time of both parallel and non parallel function using Instant in rust.

EVALUATION

To evaluate the performance, I used std::time::Instant. To measure the time for parallel and non parallel function. I then compared them to see how much of a difference parallelizing a function makes.

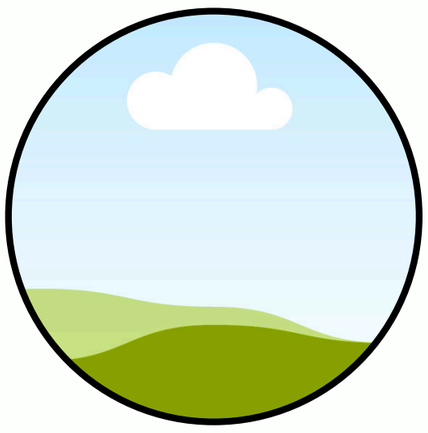




INTERESTING TIDBITS LEARNED ALONG THE WAY

I have learnt that parallelism improves performance for image processing. I also learnt that while performing image sharpening, handling the edges of the image properly is crucial to avoid artifacts.

IMAGE COMPRESSION IN RUST

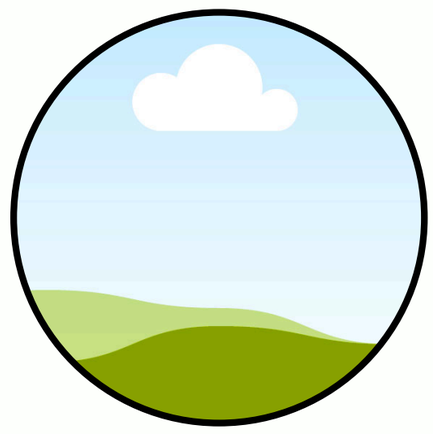


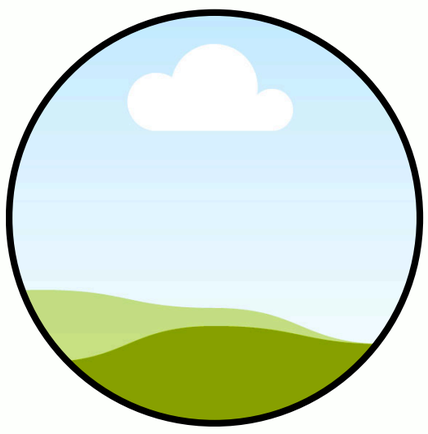
WHAT THE WORK IS ABOUT

This project demonstrates the implementation of a Rust system for compressing and converting images into JPEG format. Utilizing multicore processors, it enhances performance through parallel processing.

HOW IT IS ACCOMPLISHED

Rust: For safe and efficient system-level operations.
Rayon: For data parallelism to leverage multicore architectures.
Image Crate: Handles the decoding, encoding, and manipulation of images.



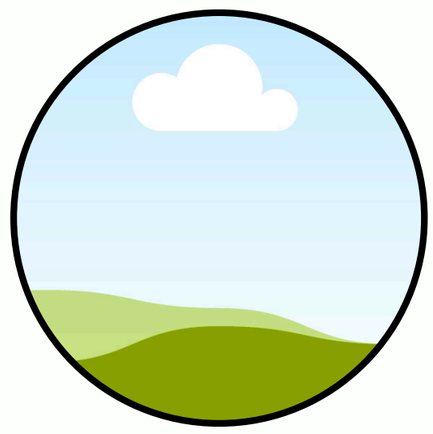


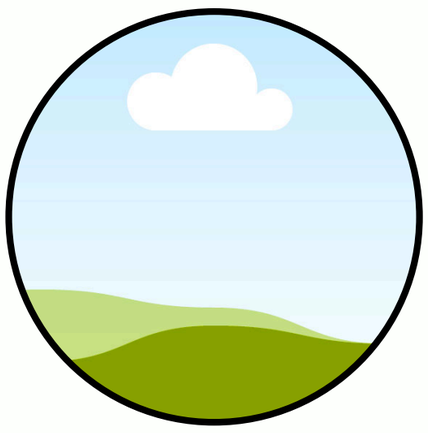
HOW IT IS ACCOMPLISHED

compress_image: Opens, decodes, and compresses a image, then saves it as JPEG.
compress_folder: Reads a directory of images, and processes each image in parallel using compress_image, saving the results in a new directory.

EVALUATION

Performance Metrics : Speedup from parallel processing
Resource Utilization : Monitors CPU and memory usage, optimizing the balance between speed and system load.

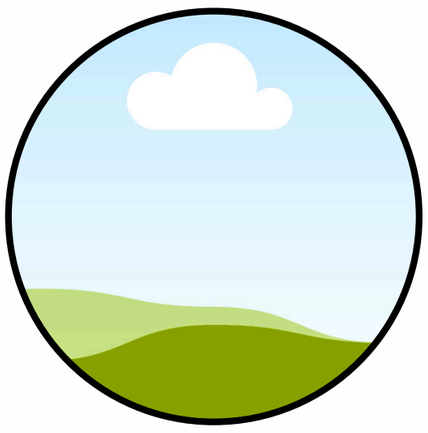




INTERESTING TIDBITS LEARNED ALONG THE WAY

Challenges of Parallel File Handling
Insights into Rust's Concurrency Model
Practical Use of External Crates : Rayon and Image

IMAGE BLURRING IN RUST

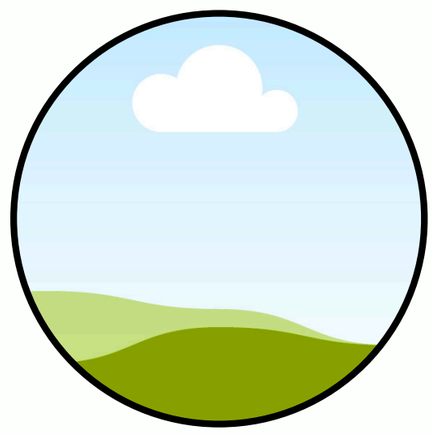


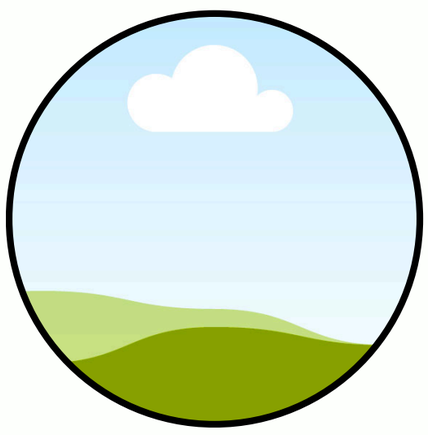
WHAT THE WORK IS ABOUT

The part focuses on image processing in rust, specifically on implementing Gaussian Blurring, it's a technique used to reduce image noise and detail. To explore parallel and sequential processing techniques for image blurring.

HOW IT IS ACCOMPLISHED

First we read the image using the get_image function, then it's decoded into a DynamicImage. Next, the image_to_chunks function is used to divide the image into smaller chunks. Next, the gaussian_blur_chunk function is used.



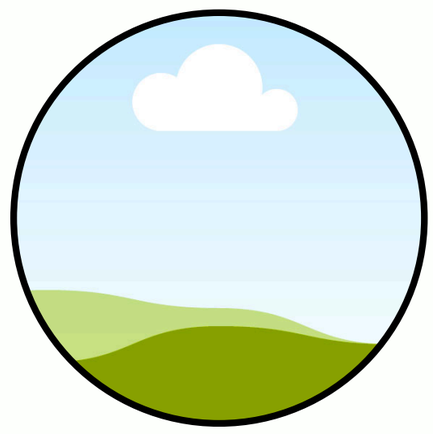


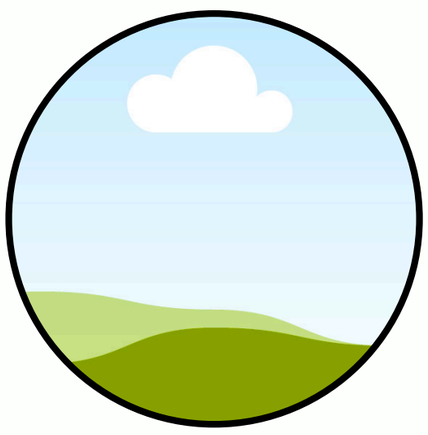
HOW IT IS ACCOMPLISHED

Next, 2 functions apply Gaussian Blur to the image chunks, one function in parallel, and the other not in parallel. Finally, the blurred chunks are reconstructed to a blurred image.

EVALUATION

To evaluate the performance, I used std::time::Instant. To measure the time for parallel and non parallel function. I then compared them to see how much of a difference parallelizing a function makes.

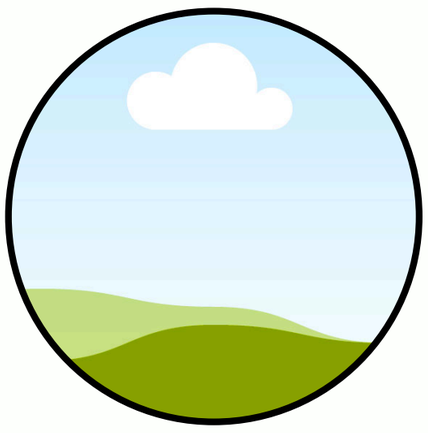




INTERESTING TIDBITS LEARNED ALONG THE WAY

I have learnt that parallelism improves performance for image processing. I also learnt that chunk processing causes issues, you need to be aware of the boundary artifacts when processing image chunks.

IMAGE SEGMENTATION RUST

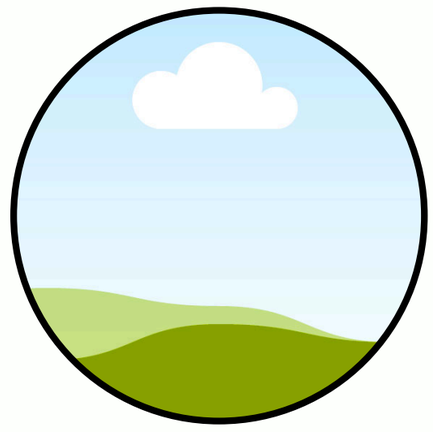


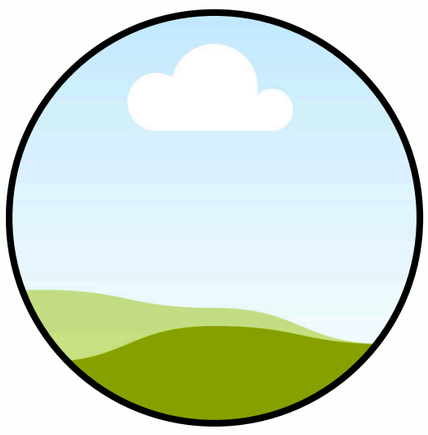
WHAT THE WORK IS ABOUT

To efficiently segment an image into clusters based on pixel colors using the K-Means clustering algorithm, implemented in Rust.

HOW IT IS ACCOMPLISHED

Parallel Processing with Rayon
NDArray Operations
Image Manipulation
Algorithm K-Mean clustering



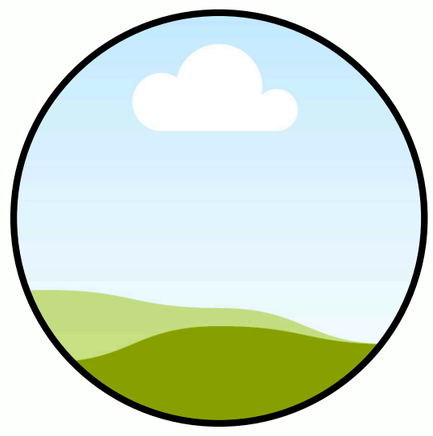


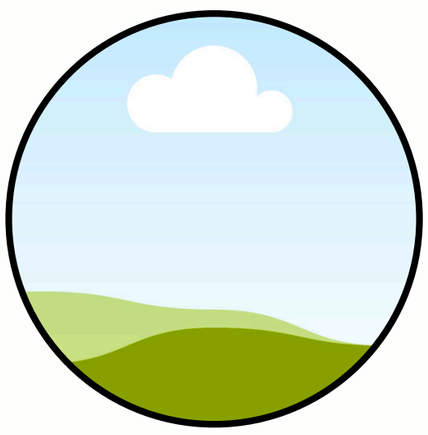
EVALUATION

Performance Metrics
Execution Time
Visual Inspection

INTERESTING TIDBITS LEARNED ALONG THE WAY

Parallel Efficiency
Algorithm Sensitivity





INTERESTING TIDBITS LEARNED ALONG THE WAY

Practical Applications : Learn K-Mean clustering in rust
Challenges in Rust