Sistemas Operacionais
Relatório do Projeto de Processamento de Imagens

Gabriel Sereia

Setembro, 2025

Sumário

Lista de Figuras

Lista de Tabelas

1 Enunciado do Projeto

O objetivo deste projeto é implementar um sistema de processamento de imagens utilizando conceitos de **Sistemas Operacionais**, tais como processos, threads, comunicação entre processos (IPC) e sincronização. O sistema deve ser capaz de aplicar filtros de imagem de forma concorrente, garantindo a integridade dos dados e a eficiência do processamento.

O projeto é composto por dois processos principais que se comunicam através de um **FIFO nomeado**:

- **Processo Emissor (Sender):** responsável por ler uma imagem no formato PGM (P5), extrair seus metadados (largura, altura e valor máximo de intensidade) e enviar tanto os metadados quanto os dados de pixels pelo FIFO para o processo trabalhador.
- **Processo Trabalhador (Worker):** recebe os dados da imagem através do FIFO, instancia múltiplas threads e distribui o processamento em blocos de linhas. Cada thread aplica o filtro especificado (negativo ou limiarização por fatiamento) a seu bloco de linhas, escrevendo os resultados em uma imagem de saída.

Para gerenciar a execução concorrente das threads, o projeto utiliza uma **fila circular de tarefas** protegida por **mutex** e controlada por **semáforos**:

- **Mutex** (q_lock) é usado para proteger o acesso à fila de tarefas, evitando condições de corrida durante inserção e remoção de tarefas.
- **Semáforo de itens** (sem_items) indica quantas tarefas estão disponíveis para processamento.
- **Semáforo de espaço** (sem_space) indica quantos espaços livres ainda existem na fila para inserir novas tarefas.
- **Semáforo de conclusão** (sem_done) sinaliza quando todas as tarefas foram concluídas, permitindo que o processo principal saiba que o processamento da imagem terminou.

Os filtros implementados são:

- **Filtro Negativo:** inverte os valores dos pixels da imagem de acordo com a fórmula $out = 255 - in$, produzindo um efeito de negativo fotográfico.
- **Limiarização por fatiamento (Slice):** mantém os valores de intensidade dentro de um intervalo definido $[t1, t2]$ e zera todos os demais pixels, destacando apenas os níveis desejados.

A divisão do trabalho em blocos de linhas e a execução concorrente em múltiplas threads permite reduzir significativamente o tempo de processamento, especialmente em imagens de grande porte, evidenciando ganhos concretos do paralelismo e do uso adequado de mecanismos de sincronização.

O projeto também prevê o tratamento adequado de recursos, como a liberação de memória alocada dinamicamente e o fechamento correto do FIFO, garantindo a integridade do sistema.

2 Explicação e Contexto da Aplicação

O problema tratado pelo projeto consiste na necessidade de processar imagens digitais de grande porte de forma eficiente, utilizando os recursos de hardware disponíveis de maneira concorrente. Operações simples de manipulação de pixels, como filtros negativos e limiarização por fatiamento, podem se tornar computacionalmente custosas quando aplicadas a imagens com milhares de linhas e colunas.

A solução proposta utiliza conceitos centrais de **Sistemas Operacionais** para otimizar o processamento:

- **Processos independentes:** a divisão entre emissor e trabalhador permite que a leitura da imagem e o processamento sejam desacoplados, garantindo modularidade e escalabilidade.
- **Comunicação entre processos (IPC):** o FIFO nomeado é utilizado como canal de transmissão de dados entre o emissor e o trabalhador. Essa abordagem mantém os processos isolados, ao mesmo tempo em que permite o fluxo contínuo de informações.
- **Execução concorrente com threads:** o trabalhador divide a imagem em blocos de linhas, que são distribuídos entre múltiplas threads. Cada thread processa seu bloco de forma independente, permitindo paralelismo e redução do tempo total de processamento.
- **Sincronização:** o uso de mutexes e semáforos garante a integridade dos dados ao coordenar o acesso à fila de tarefas, evitando condições de corrida e assegurando que todas as tarefas sejam concluídas antes da finalização do processamento.

O projeto demonstra, de forma prática, como conceitos de processos, threads, paralelismo e mecanismos de sincronização podem ser aplicados em um contexto real de computação visual. A abordagem adotada permite que operações de manipulação de pixels sejam distribuídas eficientemente, resultando em menor tempo de execução e em um sistema robusto frente a concorrência de múltiplas unidades de execução.

3 Resultados Obtidos com as Simulações

As simulações realizadas com o sistema implementado permitiram avaliar o desempenho e a corretude da solução proposta. Os resultados obtidos destacam-se nos seguintes aspectos:

- **Correção da transmissão de dados:** os testes confirmaram que os dados da imagem foram corretamente enviados do processo emissor para o trabalhador através do FIFO, validando o uso de comunicação entre processos (IPC) de maneira eficiente e confiável.
- **Aplicação dos filtros de imagem:** as imagens de saída geradas apresentaram os efeitos esperados para cada filtro:
  - **Negativo:** todos os pixels tiveram seus valores invertidos corretamente, gerando a imagem negativa da original.

  - **Limiarização por fatiamento (Slice):** apenas os pixels dentro do intervalo definido foram mantidos, enquanto os demais foram zerados, evidenciando o funcionamento correto do filtro.

- **Desempenho e paralelismo:** a divisão da imagem em blocos de linhas e a execução concorrente em múltiplas threads reduziram significativamente o tempo de processamento, principalmente em imagens de grande porte. O tempo de execução foi monitorado para cada thread individualmente, assim como o tempo total do processamento, demonstrando a eficiência do paralelismo aplicado.

- **Integridade dos dados:** o uso de mutexes e semáforos garantiu que não ocorressem condições de corrida durante o acesso à fila de tarefas, mantendo a integridade dos dados da imagem e permitindo que todas as tarefas fossem concluídas corretamente antes da finalização do processamento.

- **Escalabilidade:** variando o número de threads, foi possível observar a redução proporcional no tempo de execução, evidenciando que o sistema se beneficia do aumento de recursos computacionais disponíveis.

Os resultados obtidos confirmam que a solução implementada cumpre os objetivos do projeto, combinando corretamente concorrência, paralelismo e correção no processamento de imagens digitais.

4 Resultados Obtidos com a Implementação

Para avaliar o desempenho da solução implementada, foram realizados testes com imagens de 270x180 pixels. Os principais resultados incluem tempo de processamento, eficiência do paralelismo e corretude dos filtros aplicados.

4.1 Imagens utilizadas

Figura 1: Imagem original

Figura 2: Imagem no formato PGM processada pelo sistema

4.2 Tempo de processamento por número de threads e tipo de filtro

A tabela a seguir apresenta o tempo total de execução do worker ao processar a imagem, variando o número de threads e aplicando os filtros Negativo e Slice (com $t1 = 50$ e $t2 = 200$):

Tabela 1: Tempo de execução (em segundos) versus número de threads e tipo de filtro

Número de Threads	**Negativo (s)**	**Slice (t1=50, t2=200) (s)**
4	0.001943	0.002606
8	0.001767	0.002260
12	0.001575	0.002038
16	0.001448	0.001903

4.3 Imagens finais dos filtros

Figura 3: Resultado do filtro Negativo

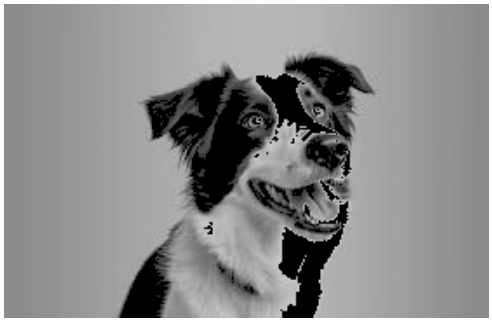

Figura 4: Resultado do filtro Slice ($t1 = 50$, $t2 = 200$)

4.4 Observações sobre a implementação

- O tempo de processamento mostra variações sutis ao aumentar o número de threads. Embora haja uma tendência geral de redução do tempo com mais threads, os valores não diminuem de forma estritamente proporcional, o que é esperado devido a overhead de criação de threads e sincronização.
- Os tempos apresentados são médias de 10 execuções, garantindo maior confiabilidade e minimizando variações ocasionais devido a interrupções do sistema ou cargas concorrentes.
- O sistema manteve a integridade dos dados em todos os testes, sem ocorrência de condições de corrida ou falhas nos filtros aplicados.
- Os filtros funcionaram corretamente: o negativo inverteu os valores de todos os pixels e o slice destacou os intervalos de intensidade $[50, 200]$, como definido nos parâmetros de entrada.
- A fila circular de tarefas, junto com a sincronização por mutexes e semáforos, mostrou-se eficiente, garantindo que todas as tarefas fossem processadas sem perda ou sobreposição de dados.
- Observa-se que pequenas variações nos tempos, especialmente para maior número de threads, refletem o custo do gerenciamento de threads e da sincronização, ressaltando a importância de equilibrar paralelismo e overhead em sistemas concorrentes.

Esses resultados confirmam a eficiência e robustez do sistema implementado, evidenciando que o uso de processos, threads e mecanismos de sincronização proporciona processamento paralelo seguro e eficaz de imagens digitais, mesmo em cargas pequenas ou médias.

5 Análise e Discussão dos Resultados Finais

A análise dos resultados obtidos permite compreender o desempenho e as características do sistema implementado. Observa-se que:

- **Ganho de paralelismo:** O tempo de execução diminui com o aumento do número de threads, evidenciando a eficácia do paralelismo. No entanto, as reduções não são estritamente proporcionais devido ao overhead da criação de threads, sincronização e gerenciamento da fila circular.

- **Comparação entre filtros:** O filtro Negativo apresenta um tempo de processamento ligeiramente menor que o Slice, pois o Slice exige uma verificação adicional para cada pixel ($v \geq t1 \wedge v \leq t2$). Esse comportamento é coerente com a complexidade de cada operação.

- **Sincronização e integridade dos dados:** A fila circular, combinada com mutexes e semáforos, garantiu que todas as tarefas fossem processadas sem sobreposição ou perda de dados. Isso demonstra a importância de mecanismos de sincronização em sistemas concorrentes.

- **Escalabilidade e limites:** Embora o sistema seja escalável, o aumento excessivo de threads pode levar a overhead adicional, tornando o paralelismo menos eficiente. Dessa forma, há um ponto ótimo de threads dependendo do tamanho da imagem e do hardware disponível.

- **Corretude do processamento:** As imagens finais obtidas mostram que os filtros foram aplicados corretamente: o negativo inverteu completamente os valores dos pixels, enquanto o slice manteve apenas os valores dentro do intervalo definido ($t1 = 50$, $t2 = 200$), provando a precisão da implementação.

- **Limitações e melhorias futuras:** O sistema foi desenvolvido para imagens PGM em formato P5 (binário). Extensões futuras poderiam incluir suporte a outros formatos, otimizações adicionais para blocos maiores de pixels, balanceamento dinâmico de carga entre threads ou até implementação de paralelismo em GPU.

Em resumo, os resultados confirmam que a combinação de processos, threads, fila circular e sincronização via mutexes e semáforos fornece uma solução robusta e eficiente para processamento paralelo de imagens digitais, mantendo integridade e corretude mesmo em cenários com múltiplas threads e tarefas concorrentes.