Remoção de Ruído de Imagem

Escala-Gray

André Zhan nº 58762
André Gonçalves nº 58392

Departamento de Informática
Arquitetura de computadores I
Docente: Miguel Barão

Conteúdo

Capítulo 1

Introdução

Este projeto, da disciplina de Arquitetura de Computadores I, tem como objetivo desenvolver um conjunto de funções em assembly RISC-V para remover ruído em imagens em tons de cizento. Desta forma o trabalho foi divido em várias partes de forma a ser possível o desenvolvimento do mesmo.
Na primeira parte do trabalho pretende-se converter a imagem de PNG para Gray utilizando o programa convert disponibilizado pelo pacote de software ImageMagick em Linux (Figura 1.1).
De seguida pretende-se filtrar a imagem de forma a realizar o denoise do mesmo, foi utilizado assim dois algortimos diferentes de forma a relizar o mesmo.

O primeiro método é o filtro de média (Figura 1.2), onde o ruído da imagem consiste em essencialmente em píxeis que foram corrompidos, apresentando intensidades adulteradas. O filtro de média consiste assim em substituir cada pixel pela média dos pixeis numa vizinhança à sua volta, sendo que os píxeis viznhos tem tipicamente intensidades parecidas, a média terá o efeito de substituir os píxeis do ruído por valores mais preto das intensidades dos píxeis vizinhos.

O segundo método é o filtro de mediana (Figura 1.3), sendo o mesmo parecido com o filtro da média porém em vez de substituir cada píxel pela média substitui pela mediana.

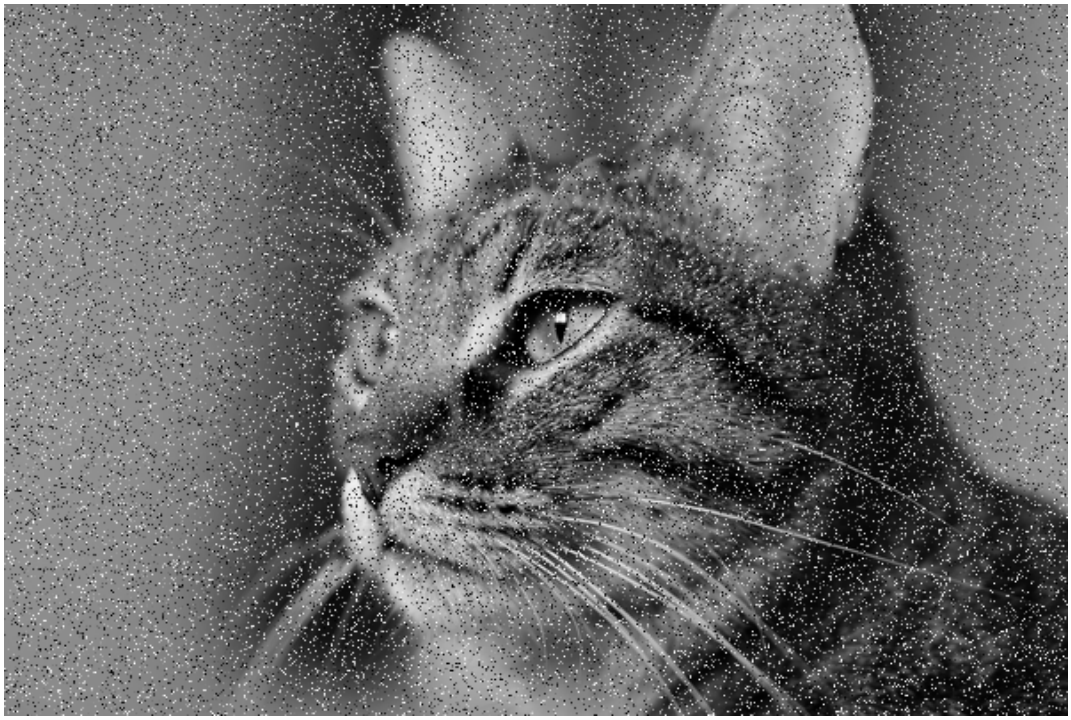

Figura 1.1: Imagem Original

Figura 1.2: Imagem filtrada
pela Escala Média.

Figura 1.3: Imagem filtrada
pela Escala Mediana.

Capítulo 2

Funções Principais

2.1 read_image

Esta função tem como objetivo abrir um arquivo de imagem, ler e armazenar os dados da imagem em um buffer na memória. Ela recebe dois parâmetros: uma string que contém o nome do arquivo a ser lido e o endereço de um buffer onde a imagem será lida.

2.2 write_image

Esta função tem como objetivo abrir um arquivo de imagem, escrever e armazenar os dados da imagem em um buffer na memória. Ela recebe dois parâmetros: uma string que contém o nome do arquivo a ser escrito e o endereço de um buffer onde a imagem será escrita.

2.3 filtro_media

Esta função tem como objetivo dar inicio ao filtro da média onde é alocado espaço na pilha para os registos "s"que vão ser utilizados e os mesmos são salvos na pilha, esta função dá também valores a alguns registos. Os registos a0, a1, a2, a3 são copiados para os registos s0, s1, s2, s3 respetivamente. Sendo a0 a entrada, a1 a saída, a2 a largura e a3 a altura. Ao registo é dado s6 o valor 9 sendo usado para definir o tamanho do array 3x3. Os registos t0, t1, t2, t3 e s4 irão ser utilizados como iteradores para os ciclos for seguintes.

2.3.1 for_y_media

A label for_y_media tem como função dar início ao loop y sobre as linhas da imagem onde controla o movimento pela altura da imagem, garantindo que todas as linhas são processadas

2.3.2 for_x_media

A label for_x_media tem como função dar início ao loop y sobre as colunas da imagem onde controla o movimento pela altura da imagem, garantindo que todas as linhas são processadas

2.3.3 for_j_media

Esta label juntamente com a label for_i_media controlam o movimento pelos píxeis vizinhos ao redor do píxel que está a ser processado. A label for_j_media é o loop sobre os pixeis vizinhos na direção vertical em relação ao pixel quie está a ser processado e o for_i_media é o loop sobre os pixeis vizinhos na direção horizontal. Esses loops garantem que a média seja calculada corretamente para os pixeis vizinhos

2.3.4 for_i_media

Esta label juntamente com a label for_j_media controlam o movimento pelos píxeis vizinhos ao redor do píxel que está a ser processado. A label for_i_media é o loop sobre os pixeis vizinhos na direção horizontal em relação ao pixel quie está a ser processado. Esses loops garantem que a média seja calculada corretamente para os pixeis vizinhos

2.4 filtro_mediana

Esta função tem como objetivo dar inicio ao filtro da mediana onde é alocado espaço na pilha para os registos "s"que vão ser utilizados e os mesmos são salvos na pilha, esta função dá também valores a alguns registos. Os registos a0, a1, a2, a3 são copiados para os registos s0, s1, s2, s3 respetivamente. Sendo a0 a entrada, a1 a saída, a2 a largura e a3 a altura. O registo t0 é dado o valor 1, para ser usado como uma constante, o registo s3 é copiado para o s5 e subtraido 1 para ser utilizado como altura -1 e o registo s2 é copiado para o registo s6 e subtraido 1 para ser utilizado como largura -1. Ao registo s7 e s8 é dado o valor 1 e os registos s9 e s11 sao dados os valores -1, todos para ser utilizados como iteradores dos ciclo for seguintes.

2.4.1 for_y_mediana

O ciclo for y dá início ao loop externo que percorre as linhas da imagem começando da segunda linha, s7 = 1.

2.4.2 for_x_mediana

O ciclo for x dá inicio ao loop interno que percorre as colunas da imagem começando da segunda colina, s8 = 1.

2.4.3 for_j_mediana/for_i_mediana

Os ciclos for j e i são os responsáveis por percorrer a janela 3x3 ao redor do pixel central estando também dentro deles o código necessário para realizar a recolha dos píxeis e a ordenação dos mesmos com a ajuda da função bubble_sort para realizar o filtro dos mesmos

Capítulo 3

Funções Auxiliares

3.1 bubble_sort

Esta função tem como propósito ordenar um array de bytes em ordem crescente utilizando o algoritmo bubble sort. No inicio alocamos espaço na pilha para os respetivos registos s a ser utilizado e de seguida utilizamos o s0 como o tamanho do array dando lhe o valor guardado em a1, e o registo s1 vai ser utilizado como o ponteiro do array. Os registos t1 e t0 são dados os valores 0 para ser utilizados como iteradores dos ciclos for seguintes.

Capítulo 4

Conclusão

Para fazer este trabalho seguimos uma ordem de etapas simples: primeiramente implementamos o código para remover o ruído de uma imagem .gray, em C, começando por implementar o filtro da média e de segyuda o da mediana. Logo chegámos à conclusão que teríamos de implementar uma função "bubblesort", que aprendemos na cadeira EDA (Estruturas de Dados e Algoritmos), para organizar um array e, por fim, obter a mediana. Implementámos a função "bubblesort"e, de seguida, a função para aplicar o filtro da mediana. Após completarmos o código em C com as restantes funções necessárias, como, por exemplo, as funções de ler/escrever a imagem, chegou a hora de escrever este código, mas em assembly, nomeadamente, RISC-V. Esta última parte foi a mais difícil para nós. Apesar do trabalho ser um pouco complexo e demorado, achamos que conseguimos alcançar o objetivo pretendido pelo professor.