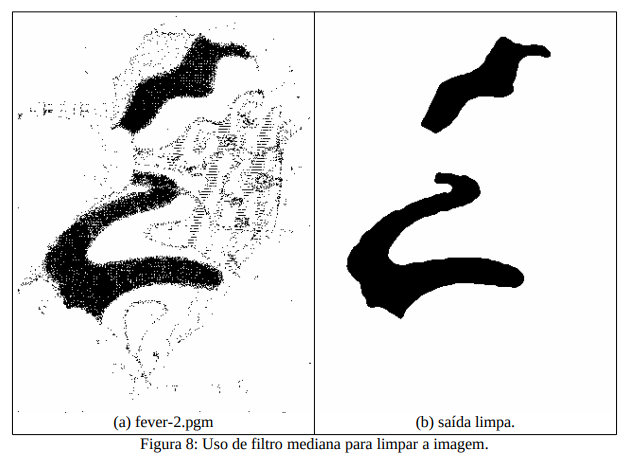
**PSI3471 - Fundamentos de Sistemas Eletrônicos Inteligentes**

Lição de Casa 2 (Aulas 3 e 4)

| **Nome Completo** | **Número USP** |
| --- | --- |
| Gustavo Henrique da Silva Amaral | 12551686 |
| Thiago da Rocha Calomino Gonçalves | 12554647 |

**Lição de casa #1 (de 2)**

Escreva um programa que usa o filtro mediana (usando a função medianBlur do OpenCV ou o filtro implementado “manualmente”) para filtrar a imagem ruidosa fever-1.pgm e fever-2.pgm (que se encontram dentro do arquivo “filtlin.zip” diretório “textura”) obtendo as imagens limpas. A figura 8 mostra a saída esperada filtrando fever-2.pgm.

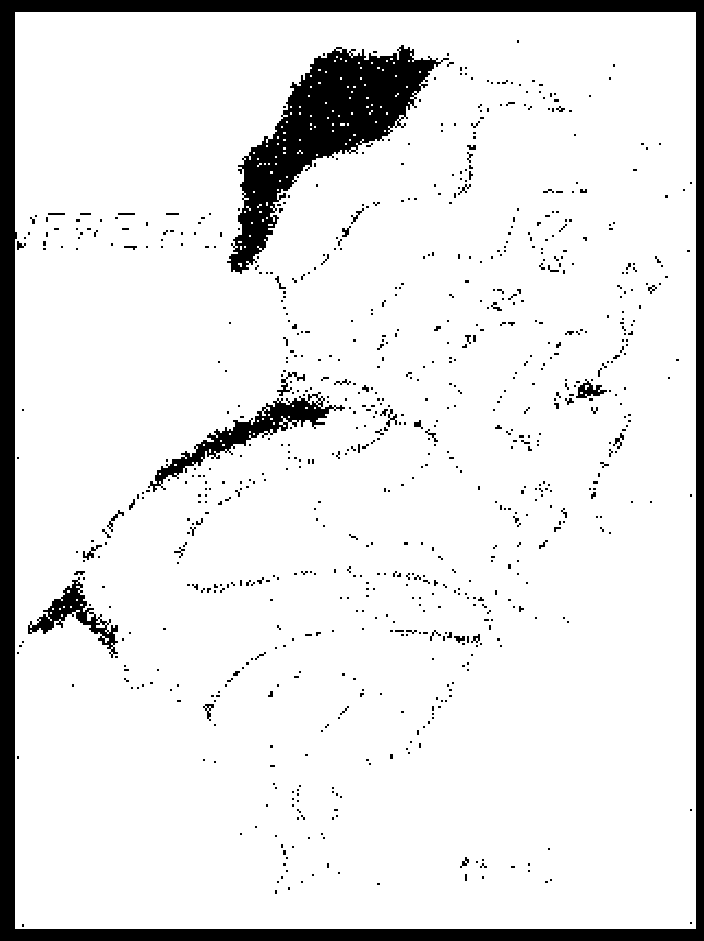


**Explicação completa:**

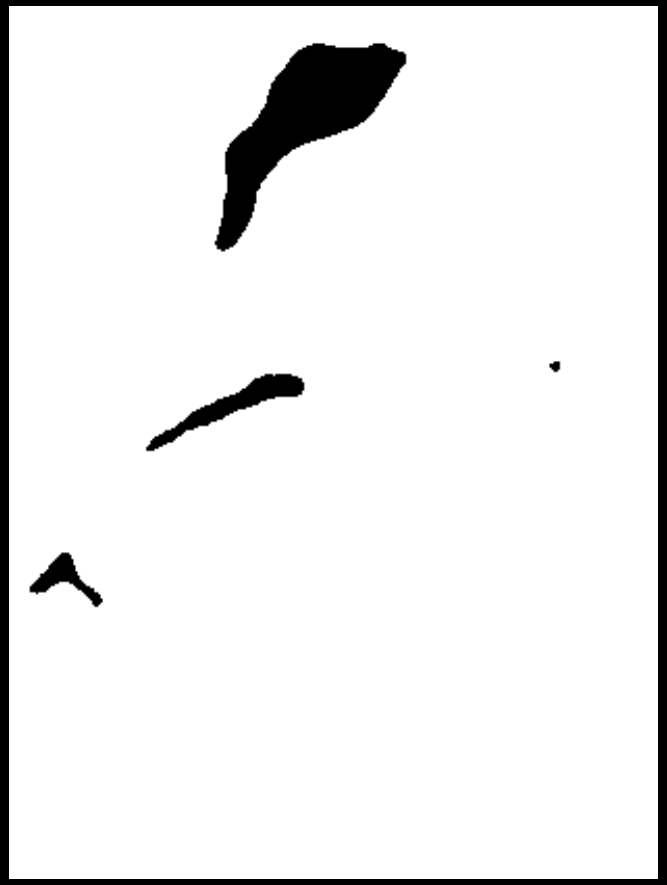
1. Leitura da Imagem: O programa carrega a imagem ruidosa *fever-2.pgm*.
2. Aplicação do Filtro Mediana:
   1. Primeiro, aplica o filtro mediana com um kernel de tamanho 3.
   2. Depois, aplica o filtro mediana com um kernel de tamanho 9.
   3. Em seguida, aplica o filtro mediana novamente com um kernel de tamanho 3 na imagem já filtrada com o kernel de tamanho 3.
3. Salvamento das Imagens: As imagens resultantes são salvas em diferentes estágios de filtragem para comparação.

O filtro mediana é bem útil no processamento de imagens para reduzir ruídos como o ruído sal e pimenta (*salt-and-pepper noise*) sem borrar excessivamente as bordas da imagem.

Abaixo segue o resultado desta aplicação sobre a imagem fever-1.pgm, como exemplo.

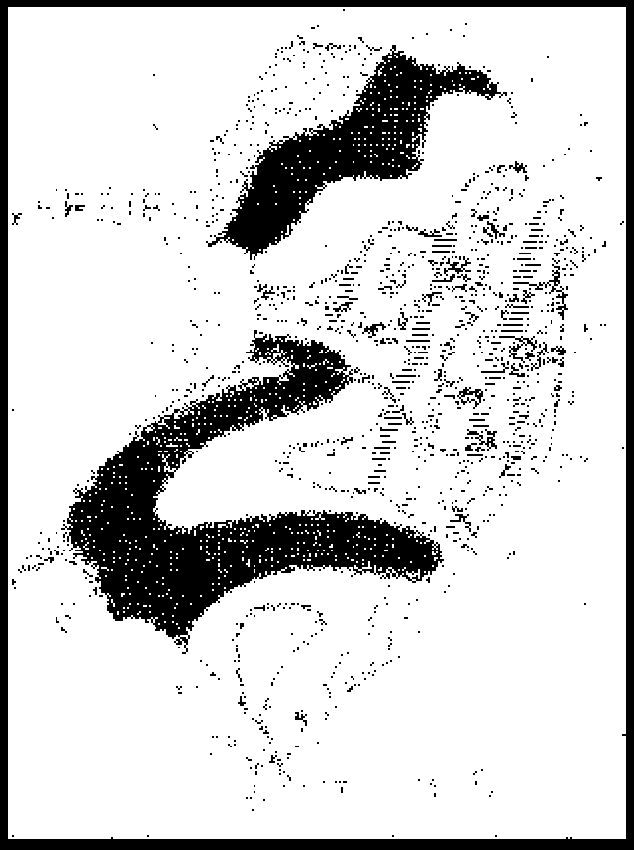


*Arquivo fever-1.pgm*

**

*Arquivos mapa2.png e mapa3.png, respectivamente*

Para o caso de fever-2.pgm:



*Arquivo fever-2.pgm*



*Arquivos mapa2.png e mapa3.png, respectivamente*

**Código da lição:**

| // Lição de casa 2 (Parte 1)  //  // Gustavo Henrique da Silva Amaral - 12551686  // Thiago da Rocha Calomino Gonçalves - 12554647  #include <cekeikon.h>  int main(){  Mat\_<GRY> a,b,c,d,e,f,g;  le(a,"fever-1.pgm");  medianBlur(a,b,3);  medianBlur(a,c,9);  medianBlur(b,d,3);  imp(a,"mapa1.png");  imp(b,"mapa2.png");  imp(c,"mapa3.png");  imp(d,"mapa4.png");  } |
| --- |

**Lição de casa #2 (de 2)**

O programa abaixo aplica filtros de Sobel 3×3 na imagem circulo.png, obtendo os componentes do gradiente ox.png e oy.png (figura 21).

//exercicio sobel.cpp - 2024

#include "procimagem.h"

int main(){

Mat\_ a = imread("circulo.png",0);

if (a.total()==0) erro("Erro leitura");

Mat\_ sx, sy, ox, oy;

Sobel(a,sx,-1,1,0,3);

ox = sx/4.0 + 128;

imwrite("ox.png",ox);

Sobel(a,sy,-1,0,1,3);

oy = sy/4.0 + 128;

imwrite("oy.png",oy);

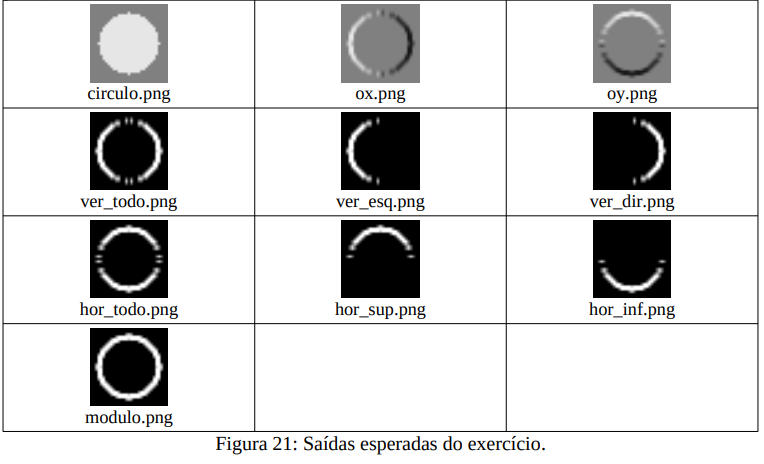
//Complete o programa aqui utilizando matrizes sx e sy

}

Complete o programa, fazendo operações de módulo pixel a pixel (abs), inverter sinal pap (- com um argumento), soma pap (+), subtração pap (- com dois argumentos), mínimo pap (min), máximo pap (max) e elevar/raiz (pow) com as matrizes sx e sy, para obter as imagens com:

* Todas as bordas verticais: ver\_todo.png
* A borda vertical esquerda: ver\_esq.png
* A borda vertical direita: ver\_dir.png.
* Todas bordas horizontais: hor\_todo.png
* A borda horizontal superior: hor\_sup.png
* A borda horizontal inferior: hor\_inf.png
* Módulo do gradiente: modulo.png

Nota: A função para elevar por 2 de C++ é pow(ent, 2, sai). Calcular a raiz quadrada é pow(ent, 0.5, sai).



**Explicação:**

Resumo das Operações

* Filtros de Sobel: Calcula os gradientes horizontal (sx) e vertical (sy).
* Normalização: Normaliza os gradientes para visualização.
* Bordas Verticais e Horizontais: Utiliza as matrizes sx e sy para detectar bordas.
* Módulo do Gradiente: Calcula a magnitude do gradiente combinando sx e sy.

Essas operações permitem identificar e visualizar as diferentes bordas na imagem circulo.png.

**Código da lição:**

| // Lição de casa 2 (Parte 2)  //  // Gustavo Henrique da Silva Amaral - 12551686  // Thiago da Rocha Calomino Gonçalves - 12554647  #include <cekeikon.h>  int main(){  Mat\_<FLT> ent,sx,sy,ox,oy, todo\_hor , hor\_sup , hor\_inf , todo\_ver , ver\_esq , ver\_dir;  le(ent,"quadrado.png");    Sobel(ent,sx,-1,1,0,3); ox=sx/4.0+0.5; imp(ox,"01ox.png");  Sobel(ent,sy,-1,0,1,3); oy=sy/4.0+0.5; imp(oy,"02oy.png");    ver\_esq = sx; imp(ver\_esq,"03ver\_esq.png");  ver\_dir = -sx; imp(ver\_dir, "04ver\_dir.png");  todo\_ver = abs(sx); imp(todo\_ver, "05todo\_ver.png");    hor\_sup = sy; imp(hor\_sup,"06hor\_sup.png");  hor\_inf = -sy; imp(hor\_inf,"07hor\_inf.png");  todo\_hor = abs(sy); imp(todo\_hor,"08todo\_hor.png");  } |
| --- |