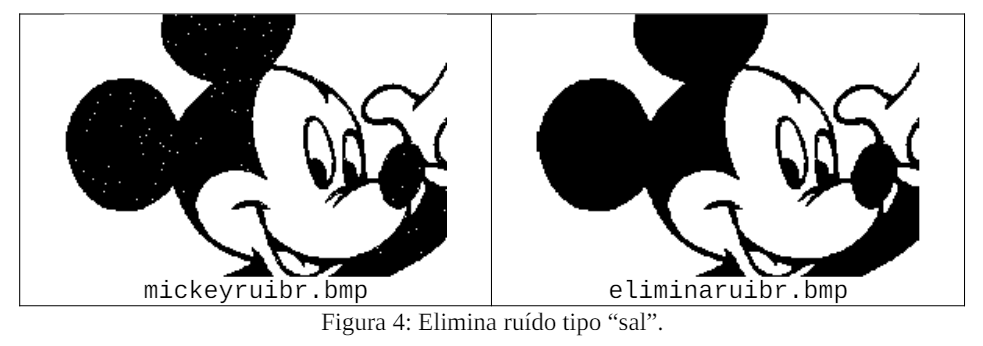
**PSI3471 - Fundamentos de Sistemas Eletrônicos Inteligentes**

Lição de Casa 1 (Aulas 1 e 2)

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

Escreva um programa que elimina o ruído branco da imagem “mickeyruibr.bmp”.

*Nota:* Você deve tomar cuidado para não acessar pixels fora do domínio da imagem (pixel antes da primeira coluna (linha) e pixel depois da última coluna (linha)), pois os valores nesses pixels estão indefinidos em C++ e pode gerar erro de índice inválido em Python.



Explicação completa:

Primeiramente, criamos a função detectaRuido(), que recebe uma matriz de intensidades de cinza (Mat\_<GRY> a) da imagem, além das coordenadas de um ponto (l, c) na imagem. Essa função retorna verdadeiro se o ponto (l, c) for considerado ruído branco e falso caso contrário.

O laço de repetição conta o número de pontos brancos ao redor de um dado ponto (l, c) na imagem. Ele percorre uma vizinhança de 3x3 centrada no ponto (l, c) e incrementa cont\_brancos sempre que encontra um ponto branco. Se o número de pontos brancos ao redor do ponto (l, c) for menor que 3, consideramos que esse ponto é ruído branco e retorna verdadeiro. Caso contrário, retorna falso.

Na função main(), inicializamos uma matriz a para armazenar nossa imagem. A função le(a, "mickey.bmp") é utilizada para carregar a imagem "mickey.bmp" na matriz a.

O duplo laço de repetição percorre todos os pixels da imagem, exceto a borda, que é ignorada.

1. Primeiro são analisados os pixels na primeira coluna da imagem. Se um pixel na primeira coluna e sua vizinhança ao redor atenderem a certas condições específicas (indicando um ponto de ruído branco), então esse pixel é marcado como preto.
2. Condições análogas são aplicadas para pixels situados na última linha da imagem.
3. Para pixels que não se encontram nas bordas da imagem, a função detectaRuido() é chamada para verificar se eles representam ruído branco. Se sim, o pixel é marcado como preto.

Por fim, a imagem processada é salva com o nome "eliminaruibr.bmp".

Código da lição:

| // Tarefa de Casa 1 - Remover Ruído Branco  // File: removeruido.cpp  #include <cekeikon.h>  bool detectaRuido(Mat\_<GRY> a, int l, int c){  int cont\_brancos = 0;  for (int i = l-1; i < l+2; i++){  for (int j = c-1; j < c+2; j++){  if (i==l && j==c){continue;}  if(a(i,j) == 255){cont\_brancos++;}  }  }  if (cont\_brancos < 3){return true;}  return false;  }  int main(){  Mat\_<GRY> a;  le(a, "mickey.bmp");  for (int l = 1; l < a.rows - 1; l++)  for (int c = 1; c < a.cols - 1; c++)  if(c == 0){ // Primeira coluna  if (a(l-1, c) == 0 && a(l-1, c+1) == 0 &&  a(l, c) == 255 && a(l, c+1) == 0 &&  a(l+1, c) == 0 && a(l+1, c+1) == 0){  a(l, c) = 0;  }  } else if (c == a.cols -1) { // Última coluna  if (a(l-1, c-1) == 0 && a(l-1, c) == 0 &&  a(l, c-1) == 0 && a(l, c) == 255 &&  a(l+1, c-1) == 0 && a(l+1, c) == 0)  {  a(l, c) = 0;  }  }  else if (l == 0) { // Primeira linha  if (a(l, c-1) == 0 && a(l, c) == 255 && a(l, c+1) == 0 && a(l+1, c-1) == 0 && a(l+1, c) == 0 && a(l+1, c+1) == 0)  {  a(l, c) = 0;  }  } else if (l == a.rows -1) { // Última Linha  if (a(l-1, c-1) == 0 && a(l-1, c) == 0 && a(l-1, c+1) == 0 &&  a(l, c-1) == 0 && a(l, c) == 255 && a(l, c+1) == 0)  {  a(l, c) = 0;  }  } else {  if (detectaRuido(a,l,c) == true) {  a(l,c) = 0;  }  }    imp(a, "eliminaruibr.bmp");  } |
| --- |

| // Tarefa de Casa 1 - Remover Ruído Branco  // File: removeruido.cpp  #include <cekeikon.h>;  bool detectaRuido(Mat\_<GRY> a, int l, int c){  int cont\_brancos = 0;  for (int i = l-1; i < l+2; i++){  for (int j = c-1; j < c+2; j++){  if (i==l && j==c){continue;}  if(a(i,j) == 255){cont\_brancos++;}  }  }  if (cont\_brancos < 3){return true;}  return false;  }  int main(){  Mat\_<GRY> a;  le(a, "mickey.bmp");  for (int l = 1; l < a.rows - 1; l++){  for (int c = 1; c < a.cols - 1; c++){    bool blackAround\_FirstColumn = a(l-1, c) == 0 && a(l-1, c+1) == 0 && a(l, c)== 255 && a(l, c+1) == 0 &&a(l+1, c) == 0 && a(l+1, c+1) == 0;  bool blackAround\_lastColumn = a(l-1, c-1) == 0 && a(l-1, c) == 0 && a(l, c-1) == 0 && a(l, c) == 255 && a(l+1, c-1) == 0 && a(l+1, c) == 0;  bool blackAround\_FirstLine = a(l, c-1) == 0 && a(l, c) == 255 && a(l, c+1) == 0 && a(l+1, c-1) == 0 && a(l+1, c) == 0 && a(l+1, c+1) == 0;  bool blackAround\_LastLine = a(l-1, c-1) == 0 && a(l-1, c) == 0 && a(l-1, c+1) == 0 && a(l, c-1) == 0 && a(l, c) == 255 && a(l, c+1) == 0;  if(c == 0){ // Primeira coluna  if (blackAround\_FirstColumn == true){a(l, c) = 0;}  } else if (c == a.cols -1) { // Última coluna  if (blackAround\_lastColumn){a(l, c) = 0;}  } else if (l == 0) { // Primeira linha  if (blackAround\_FirstLine == true){a(l, c) = 0;}    } else if (l == a.rows -1) { // Última Linha    if (blackAround\_LastLine){a(l, c) = 0;}    } else {  if (detectaRuido(a,l,c) == true) {a(l,c) = 0;}    }  }  }  imp(a, "eliminaruibr.bmp");  } |
| --- |