CI208: Programação de Computadores

Profa. Carmem Hara

Departamento de Informática/UFPR

6 de junho de 2024

Resumo

Aplicações de matrizes: processamento de imagens

Objetivos da aula

- Mostrar como manipular imagens usando matrizes

Problemas básicos com imagens

- Clarear uma imagem
- Reduzir uma imagem
- Detectar bordas

Imagens são matrizes

- Existem vários formatos para representar imagens
  - GIF, TIFF, JPEG, ...
- Vamos ver o formato PGM, que é livre, mas para tons de cinza
- Imagens coloridas ficam para um outro curso

Exemplos de imagens

O formato PGM

- O formato PGM (*Portable Graymap Format*) é um formato de imagens baseado em um arquivo ASCII
- Faz parte do formato mais geral Netpbm
  - `https://en.wikipedia.org/wiki/Netpbm`

O formato PGM

- A primeira linha contém o identificador do formato "P2"
- A segunda linha contém a largura e a altura, nesta ordem
- A terceira linha contém o valor de pixel de intensidade máxima (branco)
- O restante são os valores dos pixels (`0..255`), sendo que no final de cada linha existe um "newline"

O sapo de verdade

```
P2
121 113
255
153 153 151 148 146 144 142 141 145 152 158 161 161 156 153 150 148 147 147
147 147 146 149 148 142 137 137 139 138 138 139 139 138 137 136 136 138 139
139 138 138 138 137 135 136 145 152 145 134 131 128 126 129 131 133 135 138
144 143 142 149 155 157 159 163 168 172 179 185 182 174 169 168 170 167 165
167 166 166 166 163 164 165 165 165 165 166 164 162 163 162 162 162 160 158
158 159 157 154 155 158 162 161 159 159 159 161 162 160 159 157 153 149 145
143 140 139 141 143 144 141 153 153 153 153 151 149 147 145 145 147 150 155
160 158 155 151 148 147 146 146 146 146 147 148 145 142 141 141 141 141 139
137 136 135 135 136 137 138 138 138 138 137 139 139 138 141 145 142 132 129
131 130 128 125 127 127 126 127 128 129 132 134 136 140 145 151 151 158 170
172 169 167 166 167 169 171 171 166 162 160 158 157 160 161 160 160 160 158
158 159 161 162 162 160 157 156 157 155 153 154 157 160 160 159 159 159 160
161 161 160 159 157 152 148 148 146 142 141 138 135 137 159 159 160 160 158
etc
```

Como clarear uma imagem?

- O valor 0 significa preto e o valor da terceira linha significa branco, sendo que o valor máximo deste é 255
- Logo, para clarear uma imagem, basta somar uma constante em todos os valores da matriz, e alterar o valor do pixel de maior valor
- O processo não é sempre reversível por causa desta última alteração

Como clarear uma imagem?

- O valor 0 significa preto e o valor da terceira linha significa branco, sendo que o valor máximo deste é 255
- Logo, para clarear uma imagem, basta somar uma constante em todos os valores da matriz, e alterar o valor do pixel de maior valor
- O processo não é sempre reversível por causa desta última alteração

Como clarear uma imagem?

- O valor 0 significa preto e o valor da terceira linha significa branco, sendo que o valor máximo deste é 255
- Logo, para clarear uma imagem, basta somar uma constante em todos os valores da matriz, e alterar o valor do pixel de maior valor
- O processo não é sempre reversível por causa desta última alteração

Algoritmo para clarear

```
1  void clarear_pgm ( int O[][MAX], int l, int c, int max, int cte ){
2    int i, j;
3
4    for( i=0; i<l; i++ )
5      for( j=0; j<c; j++ ){
6        O[i][j]= O[i][j] + cte;
7        if( O[i][j] > max )
8          O[i][j]= max;
9      }
10 }
```

Como fica

História da Lena

- Lena (ou Lenna) é um ícone da área de processamento de imagens
- A imagem é usada como exemplo em um grande número de artigos científicos relevantes, pois é considerada uma imagem difícil de ser processada, dado algumas complexidades tais como sombras
- Recentemente ela foi homenageada na principal conferência mundial da área

`https://en.wikipedia.org/wiki/Lenna`

Reduzir uma imagem

- O objetivo é reduzir uma imagem à metade do seu tamanho original
- Existem várias técnicas, nós vamos utilizar a *média de 4 vizinhos sem repetição de elementos*
- Exemplo:

4	6	2	1
9	0	0	2
8	7	3	9
1	2	3	4

4	6	2	1
9	0	0	2
8	7	3	9
1	2	3	4

4.75	1.25
4.50	4.75

A ideia da solução

- O modo mais simples de implementar é primeiro visualizar como um elemento da nova imagem $D[i, j]$ é mapeado para os 4 elementos da imagem original I.

	0	1
0		
1	X	

	0	1	2	3
0				
1				
2	X	X		
3	X	X		

- Uma coordenada (i, j) é mapeada para o canto superior esquerdo em $(2 * i, 2 * j)$
- Logo, os outros elementos são os vizinhos
- $(2 * i + 1, 2 * j)$, $(2 * i, 2 * j + 1)$ e $(2 * i + 1, 2 * j + 1)$

O algoritmo

```
1  void zoom_pgm (int O[][MAX], int lO, int cO,
2                 int D[][MAX], int &lD, int &cD, int &maxD){
3
4    int i, j;
5
6    lD= lO / 2;
7    cD= cO / 2;
8    maxD= 0;
9    for( i=0; i<lD; i++ )
10     for( j=0; j<cD; j++ ){
11       D[i][j]= media_4_vizinhos(O, i, j);
12       if( D[i][j] > maxD )
13         maxD= D[i][j];
14     }
15 }
```

A função

```
1  int media_4_vizinhos (int O[][MAX], int i, int j ){
2    int x, y;
3
4   x= 2*i;
5   y= 2*j;
6   return (O[x][y] + O[x+1][y] + O[x][y+1] + O[x+1][y+1])/4;
7  }
```

Detecção de bordas

- Detecção de bordas era um passo importante no processamento de imagens
- Era preciso identificar os contornos da figura, desprezando quaisquer outros elementos não relevantes
- Exemplos:

A ideia

- Existem várias maneiras de fazer, mas a ideia é encontrar onde um conjunto de pixels muda bastante de intensidade
- Por exemplo, os vizinhos da esquerda e da direita de um certo pixel muito claro, são muito escuros, ou vice e versa
- Em uma imagem de bordas os pixels ou são brancos ou são pretos
- Usaremos o branco para a borda, o que não é borda é preto

0	2	1	3	1	2	3	2
1	1	1	50	50	50	5	4
1	2	2	50	0	50	4	2
1	1	3	50	6	50	3	1
1	1	2	50	50	50	2	4
3	4	5	3	2	1	0	2
1	4	5	2	3	5	0	1

Deteção de bordas

- Calcularemos um valor denominado *gradiente*
- O gradiente é calculado para cada pixel
- O pixel é comparado com seus vizinhos considerando-se somente as direções horizontal e vertical
- Se o valor deste pixel, multiplicado por 4, que é o número de vizinhos, for maior do que a soma destes vizinhos, significa que existe uma grande diferença entre este pixel e seus vizinhos e ele é portanto uma borda
- A diferença pode ser regulada, usando-se um *limiar* definido pelo usuário ou pelo programador

Deteção de bordas

- Calcularemos um valor denominado *gradiente*
- O gradiente é calculado para cada pixel
- O pixel é comparado com seus vizinhos considerando-se somente as direções horizontal e vertical
- Se o valor deste pixel, multiplicado por 4, que é o número de vizinhos, for maior do que a soma destes vizinhos, significa que existe uma grande diferença entre este pixel e seus vizinhos e ele é portanto uma borda
- A diferença pode ser regulada, usando-se um *limiar* definido pelo usuário ou pelo programador

Deteção de bordas

- Calcularemos um valor denominado *gradiente*
- O gradiente é calculado para cada pixel
- O pixel é comparado com seus vizinhos considerando-se somente as direções horizontal e vertical
- Se o valor deste pixel, multiplicado por 4, que é o número de vizinhos, for maior do que a soma destes vizinhos, significa que existe uma grande diferença entre este pixel e seus vizinhos e ele é portanto uma borda
- A diferença pode ser regulada, usando-se um *limiar* definido pelo usuário ou pelo programador

Deteção de bordas

- Calcularemos um valor denominado *gradiente*
- O gradiente é calculado para cada pixel
- O pixel é comparado com seus vizinhos considerando-se somente as direções horizontal e vertical
- Se o valor deste pixel, multiplicado por 4, que é o número de vizinhos, for maior do que a soma destes vizinhos, significa que existe uma grande diferença entre este pixel e seus vizinhos e ele é portanto uma borda
- A diferença pode ser regulada, usando-se um *limiar* definido pelo usuário ou pelo programador

Deteção de bordas

- Calcularemos um valor denominado *gradiente*
- O gradiente é calculado para cada pixel
- O pixel é comparado com seus vizinhos considerando-se somente as direções horizontal e vertical
- Se o valor deste pixel, multiplicado por 4, que é o número de vizinhos, for maior do que a soma destes vizinhos, significa que existe uma grande diferença entre este pixel e seus vizinhos e ele é portanto uma borda
- A diferença pode ser regulada, usando-se um *limiar* definido pelo usuário ou pelo programador

O algoritmo

```
1  void detectar_bordas_pgm( int O[][MAX], int D[][MAX], int l, int c ){
2    int i, j, grad;
3    /* extremidades recebem zero (PRETO) */
4    for( i=0; i<l; i++ ){
5      D[i][0]= PRETO;
6      D[i][c-1]= PRETO;
7    }
8    for( i=0; i<c; i++ ){
9      D[0][i]= PRETO;
10     D[l-1][i]= PRETO;
11   }
12
13   for( i=1; i<l-1; i++ )
14     for( j=1; j<c-1; j++ ){
15       grad= abs (O[i][j]*4 - (O[i-1][j] + O[i+1][j] + O[i][j-1] + O[i][j+1]));
16       if( grad > LIMIAR )
17         D[i][j]:= BRANCO;
18       else
19         D[i][j]:= PRETO;
20     }
21 }
```

Licença

- Slides feitos em LaTeX usando beamer
- Licença

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`