EFEITO BLUR EM IMAGENS .ppm

ATIVIDADE

Bruno Sant'Anna

January 20, 2024

Introdução

Uma *convolução* é uma operação matemática com muitas utilidades, inclusive em computação gráfica e processamento de imagens. Por meio de um *kernel de convolução*, que é uma matriz pequena com uma série de pesos, que mudam o valor de um pixel na imagem dependendo dos valores dos pixels ao seu redor. Nesse caso para aplicar um efeito de *blur*, estaremos usando um kernel onde a soma de todos os pesos é igual a 1, por exemplo

$$\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} \tag{1}$$

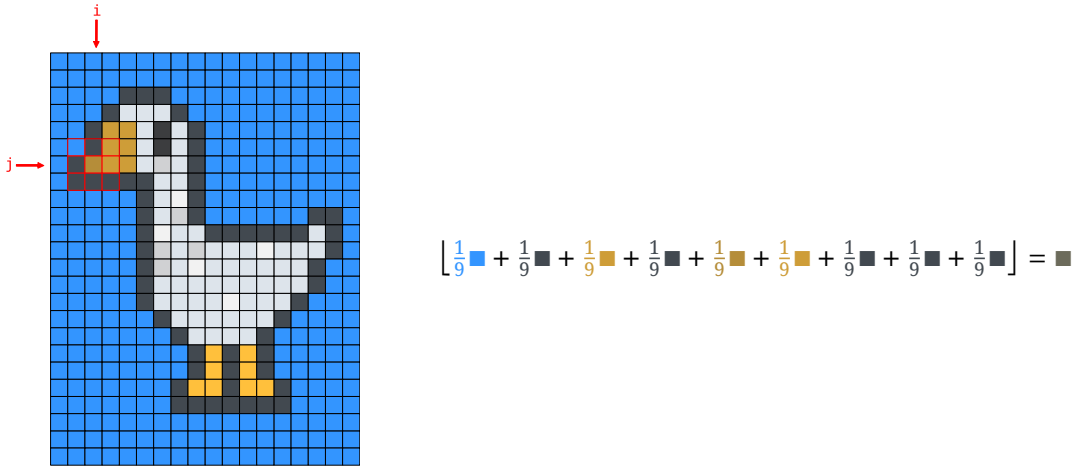
$$\begin{pmatrix} 0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\ 0.013 & 0.060 & 0.098 & 0.060 & 0.013 \\ 0.022 & 0.098 & 0.162 & 0.098 & 0.022 \\ 0.013 & 0.060 & 0.098 & 0.060 & 0.013 \\ 0.003 & 0.013 & 0.022 & 0.013 & 0.003 \end{pmatrix} \tag{2}$$

a matriz 1 aplica um blur uniforme, fazendo uma media aritmética dos pontos ao redor do centro, já a matriz 2 segue a distribuição gaussiana e aplica um peso maior ao centro.

Funcionamento

De forma simplificada, quando aplicamos um kernel em uma imagem, para cada pixel (i, j) da imagem o kernel é posicionado com o seu centro em (i, j) e multiplicamos o cada coordenada da imagem com a coordenada do kernel sobreposto e somamos, esse é o valor do pixel (i, j) na imagem nova. Vale lembrar que cada pixel da imagem colorida é um vetor de $\mathbf{N}^3$ onde cada entrada varia de 0 a 255, a primeira entrada é o canal de cor vermelho, a segunda verde e a terceira azul, conhecido como sistema RGB.

Usando o kernel 1 como exemplo:



Ou com a notação vetorial

$$\left[\frac{1}{9}\begin{bmatrix}51\\149\\255\end{bmatrix}+\frac{1}{9}\begin{bmatrix}66\\73\\80\end{bmatrix}+\frac{1}{9}\begin{bmatrix}206\\157\\56\end{bmatrix}+\frac{1}{9}\begin{bmatrix}66\\73\\80\end{bmatrix}+\frac{1}{9}\begin{bmatrix}181\\141\\57\end{bmatrix}+\begin{bmatrix}206\\157\\56\end{bmatrix}+\frac{1}{9}\begin{bmatrix}66\\73\\80\end{bmatrix}+\frac{1}{9}\begin{bmatrix}66\\73\\80\end{bmatrix}+\frac{1}{9}\begin{bmatrix}66\\73\\80\end{bmatrix}\right]=\frac{1}{9}\begin{bmatrix}108\\107\\91\end{bmatrix}$$

Resultados

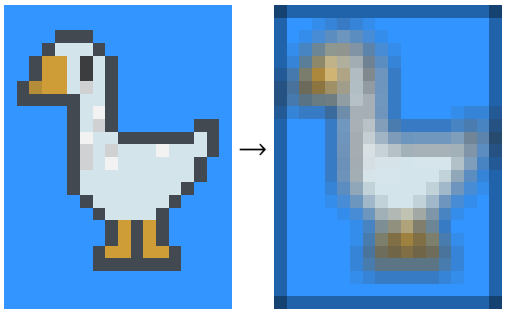


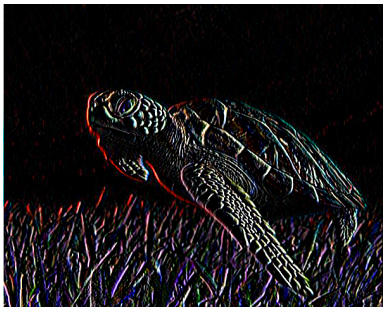
Imagem original

Blur com kernel uniforme 3×3 (eq. 1)

Blur com kernel gaussiano 5×5 (eq. 2)

Blur com kernel uniforme 9×9

Aumento de nitidez (*sharpen*) 5×5



Detecção de borda

Onde a matriz usada para o efeito *sharpen* é

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{3}$$

e para detecção de bordas

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{4}$$

Código fonte

```
void conv(int height, int length, int width, int kernelSize, unsigned char
↪  (**A)[3], float (*B)[kernelSize], int y, int x, unsigned char *output){
    int i, j, k, I, J;

    float s[3] = {0,0,0};
    int m = floor(kernelSize*0.5);
    int n = kernelSize - m;
    for (i = y-m ; i < y+n; i++){
        for (j = x-m; j < x+n; j++){
            if (i >= 0 && i < height && j >= 0 && j < length){
                I = i - (y-m);
                J = j - (x-m);
                s[0] += A[i][j][0] * B[I][J];
                s[1] += A[i][j][1] * B[I][J];
                s[2] += A[i][j][2] * B[I][J];
            }
        }
    }
    for (i = 0; i < 3; i++){
        output[i] = floor(s[i]);
    }
}

void main(){
    int i, j, k, l, h;
    unsigned char type, cmax, caractere;
    FILE *fp;

    // TROCAR NOME DO ARQUIVO AQUI
    fp = fopen("turtle.ppm", "r");
    while ((caractere=getc(fp))=='#')
        while((caractere=getc(fp))!='\n');
    ungetc(caractere,fp);

    fscanf(fp, "P%hhu\n", &type);
    while ((caractere=getc(fp))=='#')
        while((caractere=getc(fp))!='\n');
    ungetc(caractere,fp);
```

```
38
39      fscanf(fp, "%u %u\n%hhu\n", &l, &h, &cmax);
40
41
42      unsigned char (**imagem)[3];
43
44      j=l*sizeof(char*);
45      imagem = malloc(j);
46
47      j=h*3;
48      for (i=0; i<l; i++)
49          imagem[i] = malloc(j);
50
51      if(type==3){
52          for(j=0; j<h; j++)
53              for(i=0; i<l; i++)
54                  fscanf(fp, "%hhu %hhu %hhu",
                        ↪  &imagem[i][j][0],&imagem[i][j][1],&imagem[i][j][2]);
55          fclose(fp);
56      }
57      else if(type==6){
58          for(j=0; j<h; j++)
59              for(i=0; i<l; i++)
60                  fscanf(fp, "%c%c%c",
                        ↪  &imagem[i][j][0],&imagem[i][j][1],&imagem[i][j][2]);
61          fclose(fp);
62      }
63      else{
64          printf("Formato inválido!");
65          fclose(fp);
66          exit(0);
67      }
68
69      unsigned char (**blur)[3];
70      j=h*sizeof(char*);
71      blur = malloc(j);
72
73      j=h*3;
74      for (i=0; i<l; i++)
75          blur[i] = malloc(j);
76
77      unsigned char aux[3];
78
79      // KERNEL DE CONVOLUÇÃO
80      float uniform[5][5];
81      for (i = 0; i < 5; i++){
82          for (j = 0; j < 5; j++){
83              uniform[i][j] = 0.04;
84          }
85      }
86
```

```
87      double gaussian[5][5] = {
88          {0.003, 0.013, 0.022, 0.013, 0.003},
89          {0.013, 0.060, 0.098, 0.060, 0.013},
90          {0.022, 0.098, 0.162, 0.098, 0.022},
91          {0.013, 0.060, 0.098, 0.060, 0.013},
92          {0.003, 0.013, 0.022, 0.013, 0.003}
93      };
94
95      int sharpen[3][3] = {
96          {0, -1,  0},
97          {-1, 5, -1},
98          {0, -1,  0}
99      };
100
101     int bordas[3][3] = {
102         {-1, 0, 1},
103         {-2, 0, 2},
104         {-1, 0, 1}
105     };
106
107     // APLICANDO O EFEITO DE BLUR
108     for (j = 0; j < h; j++){
109         for (i = 0; i < l; i++){
110             conv(l,h,3,5,imagem,uniform,i,j,aux);
111             blur[i][j][0] = aux[0];
112             blur[i][j][1] = aux[1];
113             blur[i][j][2] = aux[2];
114         }
115     }
116
117     // EXPORTANDO A IMAGEM
118     fp = fopen("blur.ppm", "w");
119     fprintf(fp, "P6\n");
120         fprintf(fp, "%u %u\n255\n", l, h);
121         for (j=0;j<h;j++)
122             for (i=0;i<l;i++)
123                 fprintf(fp,"%c%c%c", blur[i][j][0],blur[i][j][1],blur[i][j][2]);
124         fclose(fp);
125 }
```