

## 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

$$\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} \quad (1)$$

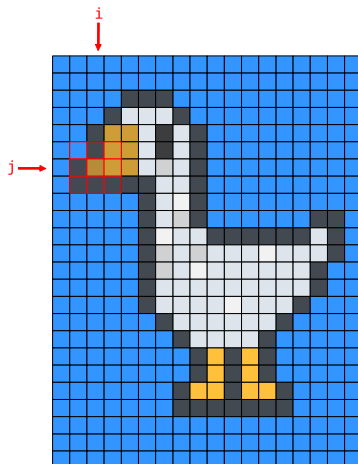
$$\begin{pmatrix} 0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\ 0.013 & 0.060 & 0.098 & 0.060 & 0.013 \\ 0.022 & 0.098 & 0.162 & 0.098 & 0.022 \\ 0.013 & 0.060 & 0.098 & 0.060 & 0.013 \\ 0.003 & 0.013 & 0.022 & 0.013 & 0.003 \end{pmatrix} \quad (2)$$

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:



$$\left[ \frac{1}{9} \text{blue} + \frac{1}{9} \text{black} + \frac{1}{9} \text{yellow} + \frac{1}{9} \text{black} + \frac{1}{9} \text{yellow} + \frac{1}{9} \text{black} + \frac{1}{9} \text{black} + \frac{1}{9} \text{black} + \frac{1}{9} \text{black} \right] = \text{black}$$

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

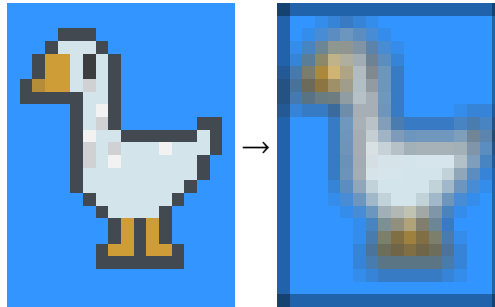


Imagem original



Blur com kernel uniforme  $3 \times 3$  (eq. 1)



Blur com kernel gaussiano  $5 \times 5$  (eq. 2)



Blur com kernel uniforme  $9 \times 9$



Aumento de nitidez (*sharpen*)  $5 \times 5$



Detecção de borda

Onde a matriz usada para o efeito *sharpen* é

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad (3)$$

e para detecção de bordas

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad (4)$$

## Código fonte

```
1 void conv(int height, int length, int width, int kernelSize, unsigned char
   ↪  (**A)[3], float (*B)[kernelSize], int y, int x, unsigned char *output){
2     int i, j, k, I, J;
3
4     float s[3] = {0,0,0};
5     int m = floor(kernelSize*0.5);
6     int n = kernelSize - m;
7     for (i = y-m ; i < y+n; i++){
8         for (j = x-m; j < x+n; j++){
9             if (i >= 0 && i < height && j >= 0 && j < length){
10                 I = i - (y-m);
11                 J = j - (x-m);
12                 s[0] += A[i][j][0] * B[I][J];
13                 s[1] += A[i][j][1] * B[I][J];
14                 s[2] += A[i][j][2] * B[I][J];
15             }
16         }
17     }
18     for (i = 0; i < 3; i++){
19         output[i] = floor(s[i]);
20     }
21 }
22
23 void main(){
24     int i, j, k, l, h;
25     unsigned char type, cmax, caractere;
26     FILE *fp;
27
28     // TROCAR NOME DO ARQUIVO AQUI
29     fp = fopen("turtle.ppm", "r");
30     while ((caractere=getc(fp))=='#')
31         while((caractere=getc(fp))!='\n');
32     ungetc(caractere,fp);
33
34     fscanf(fp, "P%hhu\n", &type);
35     while ((caractere=getc(fp))=='#')
36         while((caractere=getc(fp))!='\n');
37     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",
55                     ↪ &imagem[i][j][0], &imagem[i][j][1], &imagem[i][j][2]);
56         fclose(fp);
57     }
58     else if(type==6){
59         for(j=0; j<h; j++)
60             for(i=0; i<l; i++)
61                 fscanf(fp, "%c%c%c",
62                     ↪ &imagem[i][j][0], &imagem[i][j][1], &imagem[i][j][2]);
63         fclose(fp);
64     }
65     else{
66         printf("Formato inválido!");
67         fclose(fp);
68         exit(0);
69     }
70
71     unsigned char (**blur)[3];
72     j=h*sizeof(char*);
73     blur = malloc(j);
74
75     j=h*3;
76     for (i=0; i<l; i++)
77         blur[i] = malloc(j);
78
79     unsigned char aux[3];
80
81     // KERNEL DE CONVOLUÇÃO
82     float uniform[5][5];
83     for (i = 0; i < 5; i++){
84         for (j = 0; j < 5; j++){
85             uniform[i][j] = 0.04;
86         }
87     }

```

```

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 }

```