Conversão Matricial Algoritmos de Anti-Aliasing

Úeliton Freitas

Universidade Católica Dom Bosco - UCDB

freitas.ueliton@gmail.com

20 de outubro de 2014

Sumário

Introdução

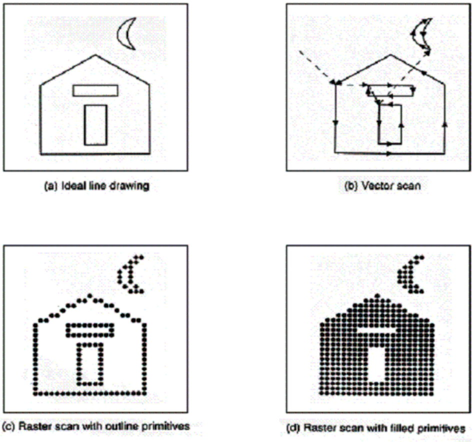


Figura : Imagen vetorial × Imagem Matricial

Introdução

Problemas

- Traçar primitivas geométricas (segmentos de retas, polígonos, circunferências, elipses, curvas,...) no dispositivo matricial.
- "rastering" $=$ conversão vetorial $\rightarrow$ matricial.
- Como ajustar uma curva, definida como coordenadas reais em um sistema de coordenadas inteiras cujos "pontos" tem área associada.

Introdução

Problemas

- Traçar primitivas geométricas (segmentos de retas, polígonos, circunferências, elipses, curvas,...) no dispositivo matricial.
- "rastering" $=$ conversão vetorial $\rightarrow$ matricial.
- Como ajustar uma curva, definida como coordenadas reais em um sistema de coordenadas inteiras cujos "pontos" tem área associada.

Introdução

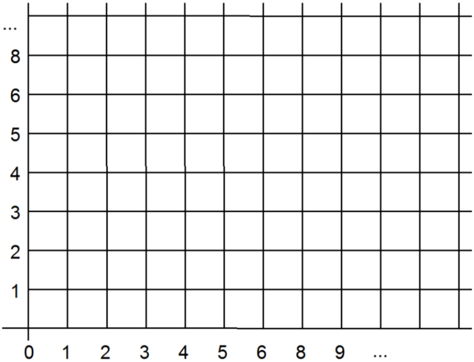


Figura : Sistema de coordenadas de dispositivo.

Conversão de Segmento de Reta

Conversão de Segmento de Reta

- Dados pontos extremos em coordenadas de dispositivo
  - $P_0(x_0, y_0)$.
  - $P_{end}(x_{end}, y_{end})$.
- Determinar quais pixels devem ser "acesos" para gerar uma boa aproximação do segmento de reta ideal.

Conversão de Segmento de Reta

Conversão de Segmento de Reta

- Características desejáveis:
  - Linearidade.
  - Precisão.
  - Espessura (Densidade Uniforme).
  - Intensidade independente de inclinação.
  - Continuidade.
  - Rapidez.

Conversão de Segmento de Reta

Equação da Reta

- Usar equação explícita da reta

$$y = m \cdot x + b$$

- m é a inclinação da reta e é dado por

$$m = \frac{y_{end} - y_0}{x_{end} - x_0}$$

- b é a intersecção do eixo y e dado por

$$b = y_0 - m \cdot x_0$$

Conversão de Segmento de Reta

Algoritmos Simples

- Varia-se x unitariamente de pixel em pixel, encontrando o valor de y.

```
{
  int x, x0, xend, y0, yend;
  float y, m;
  int valor; //cor do pixel

  m = (yend - y0)/(xend - x0);
  for (x = x0; x <= xend; x++) {
    y = y0 + m * (x - x0);
    write_pixel (x, round(y), valor); //arredonda y
  }
}
```

Conversão de Segmento de Reta

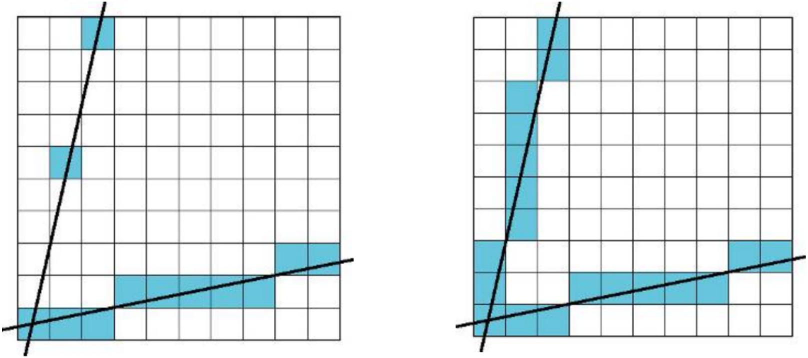
Algoritmos Simples

- Na forma dada funciona apenas com segmentos em que $0 < m < 1$. Por que?

Conversão de Segmento de Reta

Algoritmos Simples

- Se $0 < m < 1$ a variação em x é maior que em y. Caso não seja verdade, será traçado um segmento com buracos.



Conversão de Segmento de Reta

Algoritmos Simples

- Se $m > 1$ basta inverter os papéis de x e y, i.e, amostrar y em intervalos unitários, e calcular x.

$$x = x_0 + \frac{y - y_0}{m}$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Chamando δx uma variação em x, podemos encontrar a variação δy em y correspondente fazendo:

$$\delta y = m \cdot \delta x$$

- ou similarmente

$$\delta x = \frac{\delta y}{m}$$

- O algoritmo DDA se baseia no cálculo de δx e δy

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Para $|m| \leq 1$, na iteração i temos:

$$y_i = m \cdot x_i + b$$

- Sendo δ_x a variação na direção de x, na iteração $i+1$ temos:

$$\begin{aligned}
y_{i+1} &= m \cdot x_{i+1} + b \\
y_{i+1} &= m \cdot (x_i + \delta_x) + b \\
y_{i+1} &= m \cdot x_i + m \cdot \delta_x + b \\
y_{i+1} &= (m \cdot x_i + b) + m \cdot \delta_x \\
y_{i+1} &= y_i + m \cdot \delta_x
\end{aligned}$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Para $|m| \leq 1$, na iteração i temos:

$$y_{i+1} = y_i + m \cdot \delta_x$$

- se $\delta_x = 1$ então:

$$x_{i+1} = x_i + 1$$
$$y_{i+1} = y_i + m$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Se $|m| > 1$, inverte-se os papéis, isto é, $\delta_y = 1$ e calcula-se x:

$$x_i = \frac{y_i - b}{m}$$

$$x_{i+1} = \frac{y_{i+1} - b}{m}$$

$$x_{i+1} = \frac{y_i + \delta_y - b}{m}$$

$$x_{i+1} = \frac{y_i - b}{m} + \frac{\delta_y}{m}$$

$$x_{i+1} = x_i + \frac{\delta_y}{m}$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Se $|m| > 1$, inverte-se os papéis, isto é, $\delta_y = 1$ e calcula-se x:

$$x_{i+1} = x_i + \frac{\delta_y}{m}$$

- se $\delta_y = 1$, então:

$$y_{i+1} = y_i + 1$$
$$x_{i+1} = x_i + \frac{1}{m}$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Assume-se que $x_0 < x_{end}$ e $y_0 < y_{end}$ (m positivo), processando da esquerda para a direita.
- Se não é o caso, $\delta_x = -1$ ou $\delta_y = -1$, a equação deve ser adaptada.
  - Fazer as adaptações como execícios.

Digital Differential Analyzer (DDA)

Exercício

- Aplica o algoritmo de DDA para a conversão dos seguintes segmentos de retas:
  - $P_1 = (0, 1), P_2 = (5, 3)$
  - $P_1 = (1, 1), P_2 = (3, 5)$

Algoritmo de Bresenham

Algoritmo de Bresenham

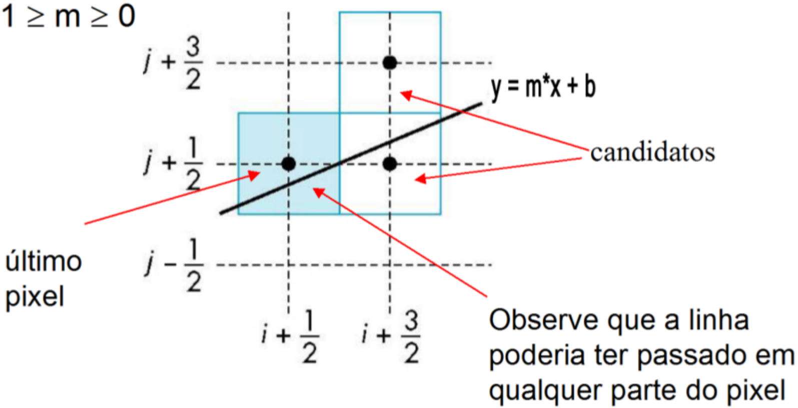
- O algoritmo DDA apesar de ser incremental, envolve cálculos com números flutuantes (cálculo de m) sendo ineficiente.
- O algoritmo de Bresenham trabalha apenas com inteiros sendo mais eficiente.

Algoritmo de Bresenham

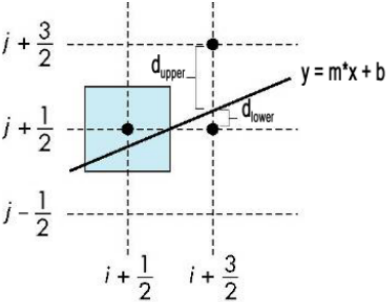
Algoritmo de Bresenham (Retas)

- Assume $0 < |m| < 1$.
- Incrementa x em intervalos unitários, calcula o y correspondente.
- Considera-se as duas possibilidades de escolha para y, decidindo qual a melhor.
  - $(x_k, y_k) \rightarrow (x_{k+1}, y_k)$
  - $(x_k, y_k) \rightarrow (x_{k+1}, y_{k+1})$

Algoritmo de Bresenham



Algoritmo de Bresenham



Algoritmo de Bresenham (Retas)

- $(d_{lower} - d_{upper} \geq 0) \rightarrow$ pixel superior.
- $(d_{lower} - d_{upper} < 0) \rightarrow$ pixel inferior.

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Com base na equação da reta ($y = m \cdot x + b$), na posição $x_k + 1$, a coordenada y é calculada da seguinte forma:

$$y = m \cdot (x_k + 1) + b$$

- Então:

$$d_{lower} = y - y_k$$
$$d_{lower} = m \cdot (x_k + 1) + b - y_k$$

- e:

$$d_{upper} = (y_k + 1) - y$$
$$d_{upper} = y_k + 1 - m \cdot (x_k + 1) - b$$

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Um teste rápido pode ser feito da seguinte forma para saber a proximidade:

$$p_k = d_{lower} - d_{upper} \quad (1)$$

$$p_k = 2m(x_k + 1) - 2y_k + 2b + 1 \quad (2)$$

- Assim:
  - $p_k > 0 \rightarrow$ pixel superior.
  - $p_k < 0 \rightarrow$ pixel inferior.

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Mas calcular m é uma operação que ainda envolve ponto flutuante:

$$m = \frac{y_{end} - y_0}{x_{end} - x_0} = \frac{\Delta_y}{\Delta_x}$$

- Substituindo m por $\frac{\Delta_y}{\Delta_x}$, e multiplicando tudo por Δ_x em (1) temos:

$$p_k = \Delta x(d_{lower} - d_{upper})$$

- Como $\Delta x > 0$, o sinal de p_k não é alterado, assim:

$$p_k = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

- com $c = 2\Delta y + \Delta x(2b - 1)$ que é um parâmetro constante e independente da posição do pixel.

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- No passo $k+1$ temos:

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c$$

- subtraindo p_k em ambos os lados temos:

$$p_{k+1} - p_k = (2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c) - p_k$$
$$p_{k+1} - p_k = 2\Delta y \cdot (x_{k+1} - x_k) - 2\Delta x \cdot (y_{k+1} - y_k)$$

- e como $x_{k+1} = x_k + 1$ (incremento unitário em x), então:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x \cdot (y_{k+1} - y_k)$$

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Nesta equação:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x \cdot (y_{k+1} - y_k)$$

- $y_k + 1 - y_k$ será 0 ou 1 dependendo do sinal de p_k
- Se $p_k < 0$, então o próximo ponto $(x_k + 1, y_k)$ então:

$$y_{k+1} - y_k = 0$$
$$p_k + 1 = p_k + 2\Delta y$$

- caso contrário o ponto será $(x_k + 1, y_k + 1)$, assim:

$$y_{k+1} - y_k = 1$$
$$p_k + 1 = p_k + 2\Delta y - 2\Delta x$$

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Este cálculo iterativo é realizado para cada posição de x começando da esquerda para a direita.
- O ponto de partida é calculado como sendo

$$p_0 = 2\Delta y - \Delta x$$

Algoritmo de Bresenham

```
void bresenham (int x1,int x2, int y1,int y2)
int dx,dy, incSup, incInf, p, x, y;
int valor;
{
  dx = x2-x1; dy = y2-y1;
  p = 2*dy-dx; /* fator de decisão: valor inicial */

  incInf = 2*dy; /* Incremento Superior */
  incSup = 2*(dy-dx); /* Incremento inferior */

  x = x1; y = y1;
  write_Pixel (x,y,valor); /* Pinta pixel inicial */

  while (x < x2) {
    if (p <= 0) { /* Escolhe Inferior */
      p = p + incInf;}
    else { /* Escolhe Superior */
      p = p + incSup;
      y++;} /* maior que 45o */
    x++;
    write_pixel (x, y, valor);
  } /* fim do while */
} /* fim do algoritmo */
```

Algoritmo de Bresenham

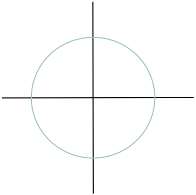
Exercício

- Aplique o algoritmo para a reta composta pelos seguintes pontos em seu extremo:
  - $P_0(0, 0)$
  - $P_1(4, 3)$

Algoritmo de Bresenham

Traçado de Circunferências

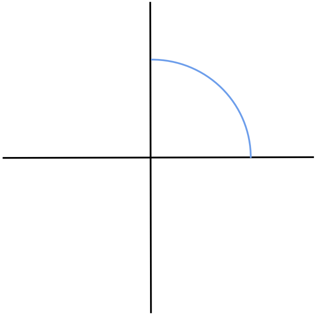
- Circunferências possuem um cento e um raio R.
- Forma explícita:
$$x^2 + y^2 = R^2$$
- Forma paramétrica:
$$x = R \cdot cos\theta, y = R \cdot send\theta$$



Algoritmo de Bresenham

Traçado de Circunferências

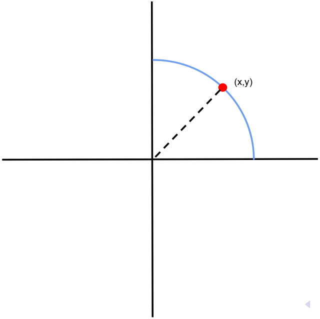
- Por que não usar a equação paramétrica?
- Por que não usar a equação explícita para traçar $\frac{1}{4}$ da circunferência?



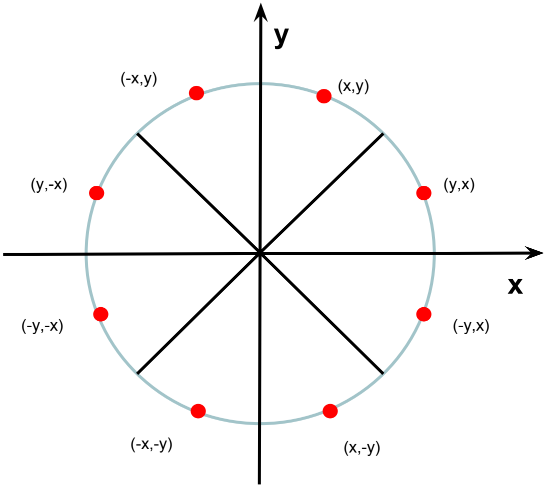
Algoritmo de Bresenham

Traçado de Circunferências - Algoritmo do Ponto Médio

- Traçando um arco de 45° no segundo octante, de $x = 0$ a $x = y = R/\sqrt{2}$
- O restante da curva pode ser obtida por simetria:
  - Se o ponto (x, y) pertence a circunferência, os outros 7 pontos podem ser obtidos por simetria de forma trivial



Algoritmo de Bresenham



Algoritmo de Bresenham

```
void CirclePoints (int x, int y, int value)
{
  write_pixel( x, y,value);
  write_pixel( x,-y,value);
  write_pixel(-x, y,value);
  write_pixel(-x,-y,value);
  write_pixel( y, x,value);
  write_pixel( y,-x,value);
  write_pixel(-y, x,value);
  write_pixel(-y,-x,value);
}
```

Figura : Simetria de Ordem 8.

Algoritmo de Bresenham

Traçado de Circunferências - Algoritmo do Ponto Médio

- Define um parâmetro de decisão para definir o pixel mais próximo da circunferência.
- Como a equação da circunferência é não linear, raízes quadradas são utilizadas para encontrar alguns pontos para o calculo das distâncias.
  - Bresenham evita as raízes comparando o quadrado das distâncias.
- Baseado na equação da circunferência, defini-se qual o pixel mias próximo da mesma.
  - Isto é feito em um único octante, o restante é feito por simetria.

Algoritmo de Bresenham

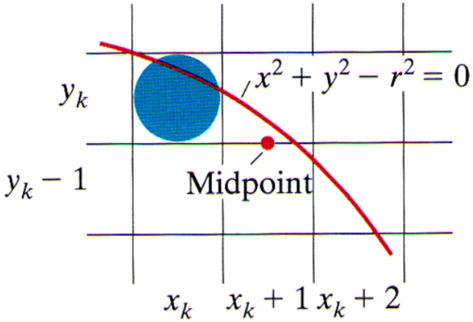
Traçado de Circunferências - Algoritmo do Ponto Médio

- $F_{circ}(x, y) = x^2 + y^2 - R^2$
  - $F_{circ}(x, y) < 0$ se (x, y) está dentro da circunferência.
  - $F_{circ}(x, y) = 0$ se (x, y) está na circunferência
  - $F_{circ}(x, y) > 0$ se (x, y) está fora da circunferência
- Incrementa x e testa o pixel que está mais perto da circunferência.
  - $F_{circ}(x, y)$ é o parâmetro de decisão e cálculos incrementais podem ser feitos.

Algoritmo de Bresenham

Traçado de Circunferências - Algoritmo do Ponto Médio

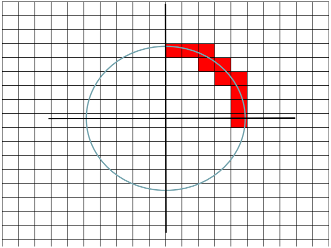
- Partindo de (x_k, y_k), as opções são:
  - $(x_k + 1, y_k)$
  - $(x_k + 1, y_k - 1)$



Algoritmo de Bresenham

Traçado de Circunferências - Algoritmo do Ponto Médio

- A função de decisão é:
  - $p_k = F_{circ}(x_k + 1, y_k - \frac{1}{2})$
  - $p_k = (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - R^2$
- se $p_k < 0$ o ponto está dentro da circunferência e y_k está mais próximo da borda.
  - Caso contrário, $y_k - 1$ está mais próximo.



Algoritmo de Bresenham

Traçado de Circunferências - Algoritmo do Ponto Médio

- A função incremental pode ser feita avaliando $x_{k+1}+1$

$$p_{k+1} = F_{circ}\left(x_{k+1}+1, x_{k+1}-1/2\right)$$
$$p_{k+1} = \left(x_{k+1}+1\right)^2 + \left(y_{k+1}-1/2\right)^2 - R^2$$
$$p_{k+1} = p_k + 2(x_k+1) + (y_{k+1}^2 y_k^2)(y_{k+1} y_k) + 1$$

- Se $p_k < 0$, então o próximo ponto é (x_{k+1}, y_k)

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

- Caso contrário será (x_k+1, y_k-1)

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$
$$\text{com } 2x_{k+1} = 2x_k + 2 \text{ e } 2y_{k+1} = 2y_k - 2$$

Algoritmo de Bresenham

Traçado de Circunferências - Algoritmo do Ponto Médio

- O valor de p_0 pode ser encontrado da seguinte forma:
  - Em p_0 temos $p_0 = F_{circ}(0, R - \frac{1}{2})$:

$$p_0 = (0+1)^2 + (R - 1/2)^2 - R^2$$
$$p_0 = 1 + (R - 1/2)^2 - R^2$$
$$p_0 = 5/4 - R$$

Algoritmo de Bresenham

```
void meanPointCirc(int cx, int cy, int R){
    x = 0;
    y = R;
    p = 4.0/5.0 - R;

    while(x<y){
        plotPixel(cx,cy,x,y)
        if(p < 0){
            p = p + 2*x+3;
        }
        else{
            y--;
            p = p + 2*x - 2*y + 5;
        }
        x++;
    }
}
```

Algoritmo de Bresenham

Traçado de Circunferências - Algoritmo do Ponto Médio

- Exercício:
  - Aplique o algoritmo do ponto médio sobre a circunferência com centro na origem e $R = 4$.

Correção de Traçado (Anti-Aliasing)

Anti-Aliasing

- Segmentos de retas em sistemas *raster* tem espessura - ocupam área.
- Devido ao processo de amostragem (discretização), segmentos de retas podem apresentar uma aparência serrilhada.

Uma possível solução

- Uma forma de reduzir este problema é utilizar pixels menores nos monitores.
    - Problemas para manter taxa de restauro em 60Hz.

Algoritmo de Bresenham

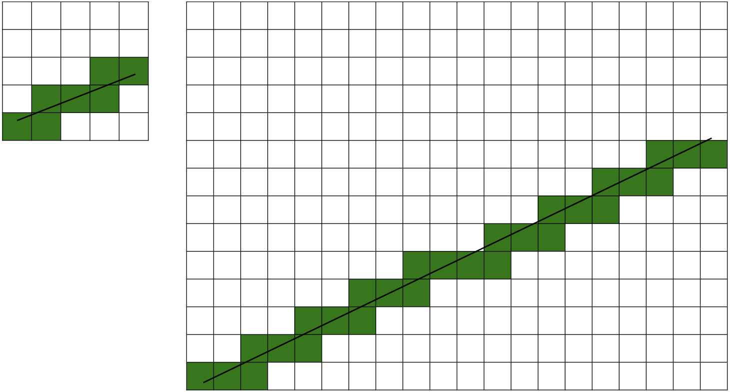


Figura : Sub-Pixels de um monitor.

Correção de Traçado (Anti-Aliasing)

Anti-Aliasing

- Em sistemas onde é possível mostrar mais de dois níveis de intensidades, é possível utilizar uma solução de software.

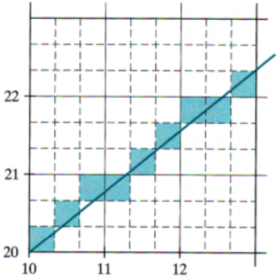
Super Amostragem

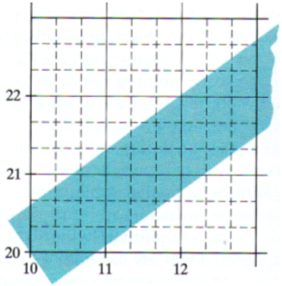
- Uma solução simples é a **superamostragem**.
  - Aumenta a taxa de amostragem simulando um monitor com um (sub) pixel de menor tamanho.
  - A intensidade do pixel é definida com base na quantidade de subpixels cobertos.

Correção de Traçado (Anti-Aliasing)

Super Amostragem

- Dividir cada pixel em sub-pixels.
  - A intensidade é dividida pelo número de sub-pixels que estão sob o caminho da linha.





Correção de Traçado (Anti-Aliasing)

Mascara de Ponderação do sub-pixel

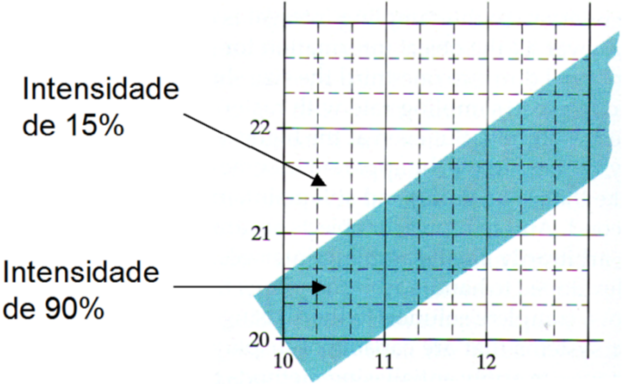
- Define uma máscara que assinala maior peso (intensidade) para sub-pixels no centro da área do pixel.

1	2	1
2	4	2
1	2	1

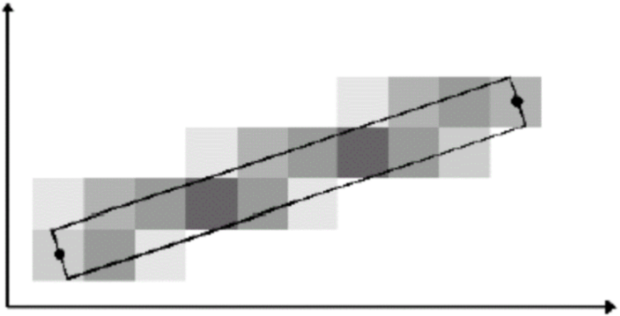
Correção de Traçado (Anti-Aliasing)

Anti-Aliasing baseado em área

- Intensidade proporcional a área coberta do pixel considerando que a linha tem largura finita.



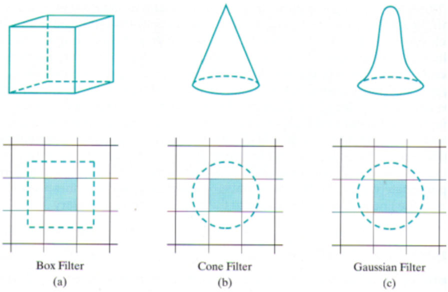
Correção de Traçado (Anti-Aliasing)



Correção de Traçado (Anti-Aliasing)

Técnicas de Filtragem

- Similar a técnica de máscara, porém mais precisa.
  - Um superfície contínua de ponderação é utilizada para determinar a cobertura do pixel.
  - A ponderação é calculada por integração - Uso de tabelas para acelerar o processo.



Correção de Traçado (Anti-Aliasing)

Compensando Diferenças de Intensidades

- O anti-aliasing pode apresentar outro problema em sistemas raster
  - Linhas desenhadas com as mesmas quantidades de pixels podem apresentar tamanhos diferentes - Linhas menores mais brilhantes.