Primitivas de Saída Gráfica - Traçado de Retas

Regina Célia Coelho

Primitivas de Saída

- Pontos e segmentos de linha são os componentes geométricos mais simples de uma figura.
- Outras primitivas de saída comuns são os círculos, superfícies cônicas, superfícies quadráticas, curvas e superfícies *splines*.
- Figuras complexas requerem grande quantidade de retas, portanto a velocidade é importante.

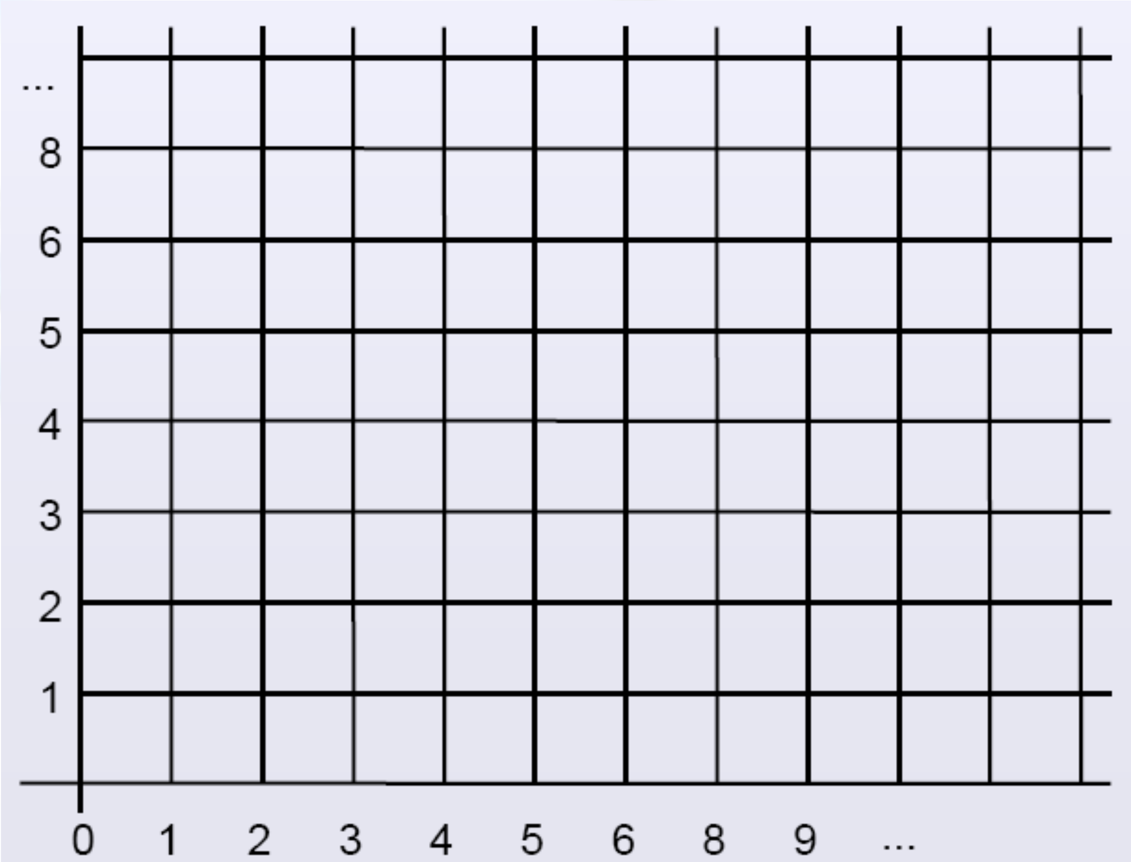
Primitivas de Saída *(cont.)*

- Como traçar primitivas geométricas (segmentos de reta, polígonos, circunferências, elipses, curvas, ...) no dispositivo matricial?

- '*rastering*' = conversão vetorial -> matricial

- Como ajustar uma curva, definida por coordenadas reais em um sistema de coordenadas contínuo, a uma malha de coordenadas inteiras?

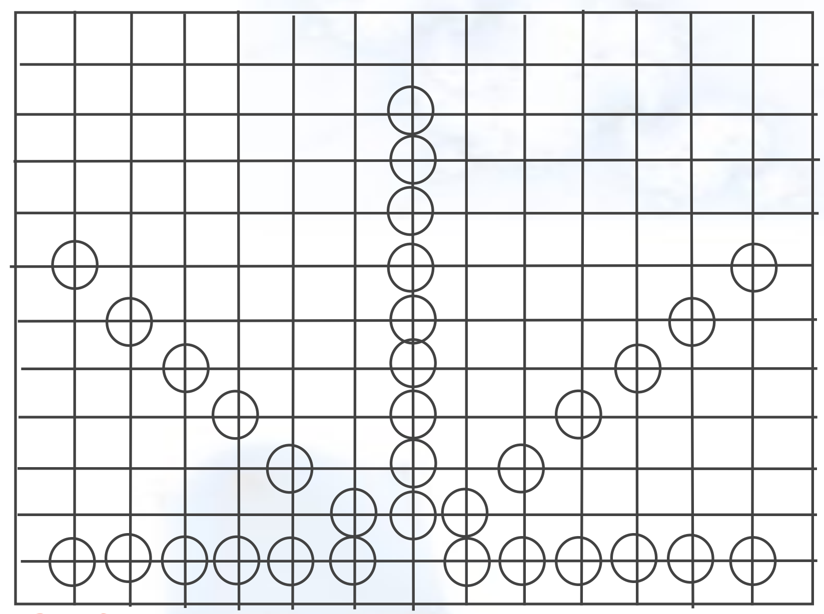
Sistema de Coordenadas do Dispositivo



Traçado de retas

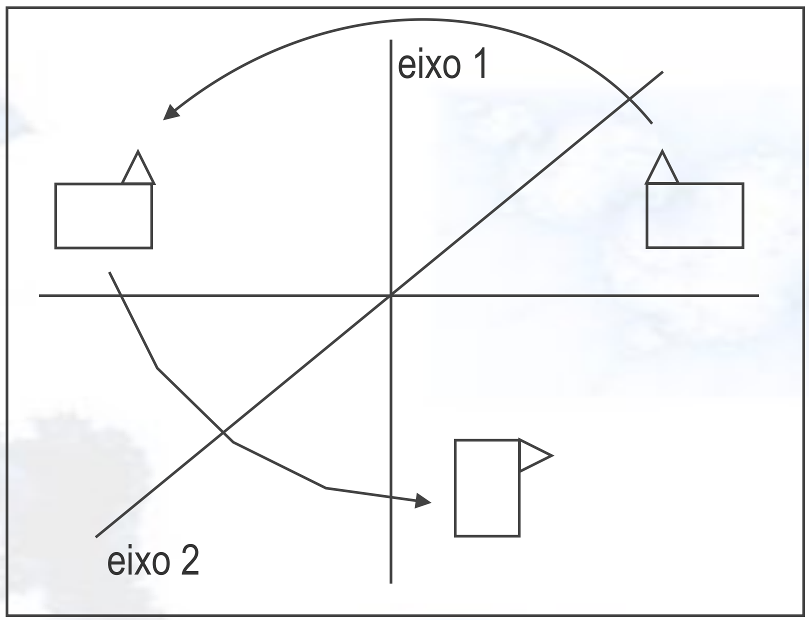
- Traçar uma reta pode ser trivial.

- O traçado de retas horizontais, verticais e diagonais a 45º e 135º não apresenta deformações.



Traçado de retas *(cont.)*

- Simetria é amplamente explorada no traçado de retas.



Traçado de retas *(cont.)*

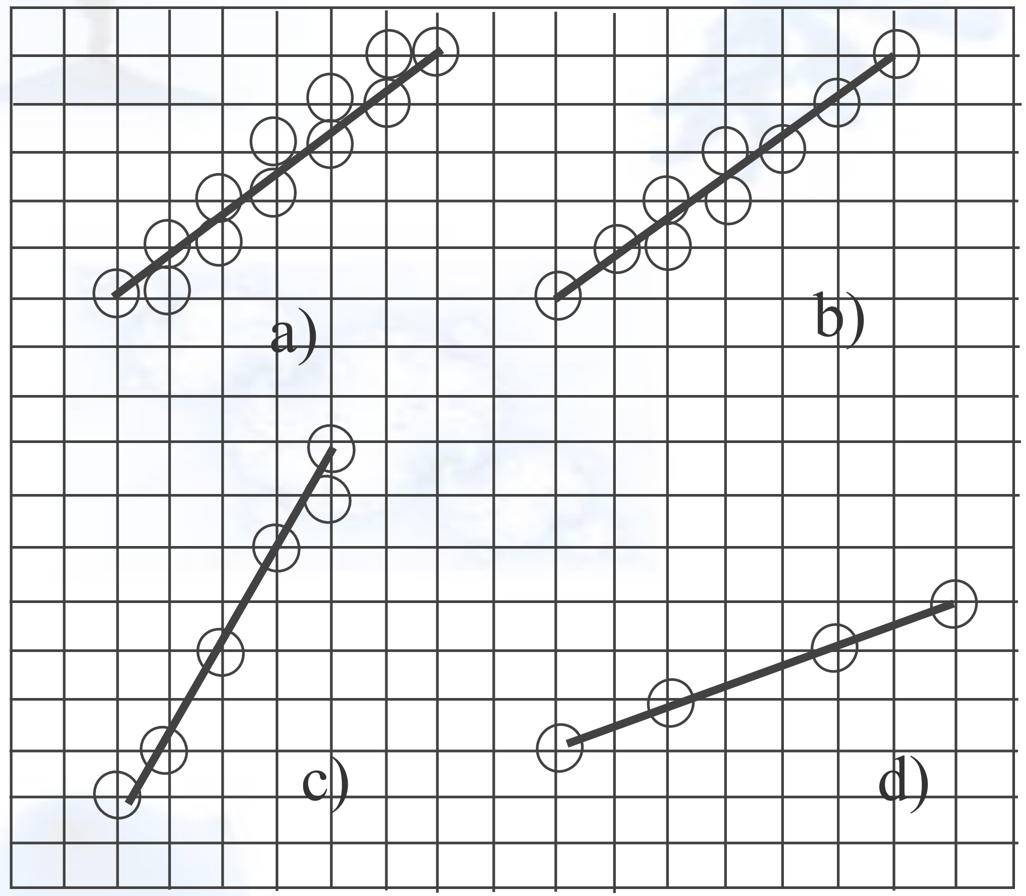
- Alguns critérios para conversão matricial de retas:

a) seleciona-se 2 pixels imediatamente acima e abaixo do ponto de interseção, a menos quando o segmento passa por interseção.

b) seleciona todos os pixels com coordenadas obtidas por arredondamento (em 45º os resultados em a) e b) são semelhantes).

c) seleciona em cada vertical o pixel mais próximo do ponto de interseção. Restrição: descontinuidade.

d) mesmo que em c), só que para linhas horizontais. Restrição: descontinuidade.



Regina Célia Coelho

Traçado de retas *(cont.)*

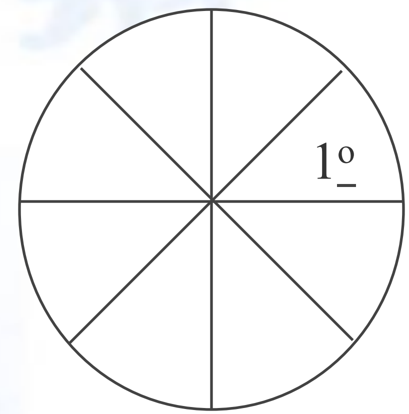
- Qualquer algoritmo pode ser testado em relação aos requisitos anteriores:
  - ✓ **linearidade**: calcula-se o soma dos quadrados das diferenças entre cada par (x_1,y_1) e (x_2,y_2). O melhor algoritmo é o que resultar na menor diferença.
  - ✓ **precisão**: problemático apenas quando os pontos extremos não são especificadas em coordenadas da tela.
  - ✓ **espessura**: (Nº de *pixels* traçados)/(comprimento da linha).
  - ✓ **intensidade e rapidez**: avaliados pela execução de *benchmarks* em cada algoritmo para efeito de comparação.
  - ✓ **continuidade**: analisa-se o tamanho dos "buracos" entre pontos, o que pode ser feito calculando a distância entre 2 pontos.

Traçado de retas (cont.)

- Problema a se resolver:
  - Dados pontos extremos em coordenadas do dispositivo:
    - P1(x1,y1) (inferior esquerdo)
    - P2(x2,y2) (superior direito)
  - Determinar quais *pixels* devem ser traçados para gerar na tela uma boa aproximação do segmento de reta ideal.

Algoritmo DDA (Digital Differential Analyzer)

- Seja uma reta definida por seus extremos (x_1,y_1) e (x_2,y_2).
- Supondo que a reta esteja no 1º octante, teremos:



$$0 < x_1 < x_2$$
$$0 < y_1 < y_2$$
$$y_2 - y_1 < x_2 - x_1$$

- Assim, a reta cortará mais linhas verticais que horizontais.
- O algoritmo deverá considerar apenas o *pixel* mais próximo da interseção da reta com a vertical.

Algoritmo DDA *(cont.)*

- A estratégia mais simples usa equação explícita da reta:

$$y = mx + b$$

$$m = (y_2 - y_1)/(x_2 - x_1) \text{ /*inclinação */}$$

$$b = y_1 - m{*}x_1 \quad \text{/* intersecção eixo y */}$$

$$y = mx + (y_1 - m{*}x_1) = y_1 + m{*}(x-x_1)$$

Algoritmo DDA (cont.)

- Algoritmo:

```
{
    int x, x1, x2, y1, y2;
    float y, m;
    int valor;
    m = (y2-y1)/(x2-x1);    /* 0 < m < 1 */
    for (x = x1; x <= x2; x++) {
        y= y1+ m*(x-x1);
        write_pixel (x, round(y), valor);   /* arredonda y */
    }
}
```

Algoritmo DDA (cont.)

- Algoritmo:

```
{
    int x, x1, x2, y1, y2;
    float y, m;
    int valor;
    m = (y2-y1)/(x2-x1);    /* 0 < m < 1 */
    for (x = x1; x <= x2; x++) {
       y= y1+ m*(x-x1);
       write_pixel (x, round(y), valor);   /* arredonda y */
     }
}
```

✓ pouco eficiente → utiliza cálculo em ponto flutuante

Algoritmo DDA - Otimização

- Na iteração *i*:

$$y_i = mx_i + B$$

- Na iteração *i+1*:

$$y_{i+1} = mx_{i+1} + B = m(x_i + \Delta_x) + B$$

$$= mx_i + m\Delta_x + B = y_i + m\Delta_x$$

- Se $\Delta_x = 1$, então

$$x_{i+1} = x_i + 1, \quad \text{e} \quad y_{i+1} = y_i + m$$

- Algoritmo incremental!!

Algoritmo DDA - Otimização

- Na forma dada, funciona para segmentos em que $0 < m < 1$.

- Por que?

Exercício

- Aplique o algoritmo (e adaptações) para encontrar os pontos dos seguintes segmentos de reta:
  - ✓P1: (0,1)   P2: (5,3)
  - ✓P1: (1,1)   P2: (3,5)

Algoritmo DDA *(cont.)*

- Funciona se $0 < m < 1$, i.e., assume que a variação em x é superior à variação em y. Se esse não for o caso, vai traçar um segmento com buracos!!

- Se $m > 1$, basta inverter os papéis de x e y, i.e., amostra y a intervalos unitários, e calcula x.

Algoritmo DDA (cont.)

- Assume $x_1 < x_2$ e $y_1 < y_2$ (m positivo), processamento da esquerda para a direita.

- Se não é o caso, então $Dx = -1$ ou $Dy = -1$, e a equação de traçado deve ser adaptada de acordo.

- Exercício: Fazer a adaptação em cada caso.

Algoritmo do Ponto-Médio ou Bresenham

- Utiliza apenas variáveis inteiras, permitindo que o cálculo de (x_{i+1}, y_{i+1}) seja feito incrementalmente.

- Assume inclinações de retas entre *0* e *1* (outras inclinações podem ser obtidas por simetria).
  - ✓ Incrementa *x* em intervalos unitários e calcula o y correspondente

- O ponto (x_1, y_1) é sempre o inferior esquerdo e o (x_2, y_2) o superior direito.

Algoritmo do Ponto-Médio ou Bresenham

- Supondo que (x_p,y_p) seja o *pixel* que acabou de ser traçado.

✓ O próximo deve ser escolhido entre *NE* e *E*;

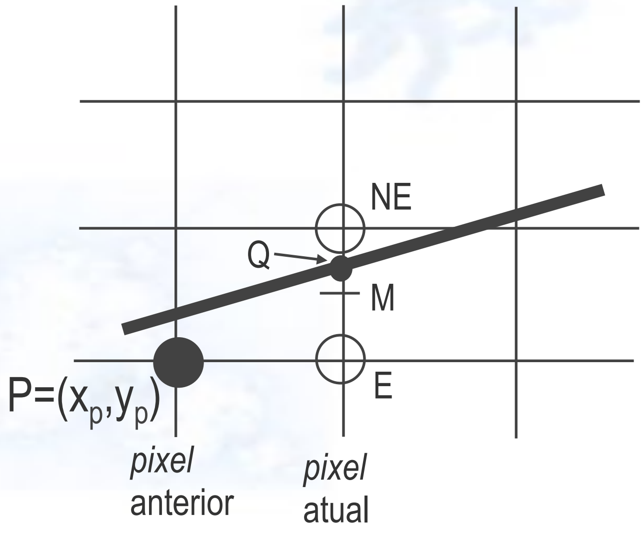
✓ Q é a interseção entre a reta e a coluna $x = x_{p+1}$;

✓ M é o ponto médio entre *NE* e *E*;

✓ Observa-se de que lado da reta está M (acima ou abaixo);

✓ Se *M* está acima, o pixel *E* está mais próximo da reta, senão o *NE* está mais próximo;

✓ O teste do ponto médio permite a escolha do pixel mais próximo da reta.



Algoritmo do Ponto-Médio ou Bresenham

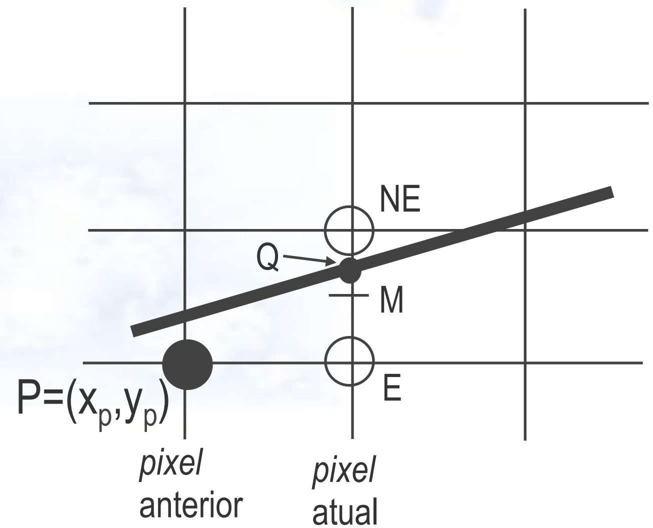
- O erro (distância entre o *pixel* escolhido e a reta) é sempre ≤ $^1/_2$
- Método para calcular de que lado está *M*:
  - Considere a função implícita: $F(x,y) = ax + by + c = 0$
  - Se $dy = y_2 - y_1$ e $dx = x_2 - x_1$, a eq. da reta em termos de sua inclinação será:

$$y = \frac{dy}{dx} x + B$$

  - Igualando a equação anterior a zero teremos: $dy*x - dx*y + B*dx = 0 = F(x,y)$
  - Assim, teremos $a = dy$, $b = -dx$ e $c = B*dx$
  - $F(x,y) < 0$ ⇨ estamos acima da linha (ponto médio acima da reta)
  - $F(x,y) > 0$ ⇨ estamos abaixo da linha (ponto médio abaixo da reta)

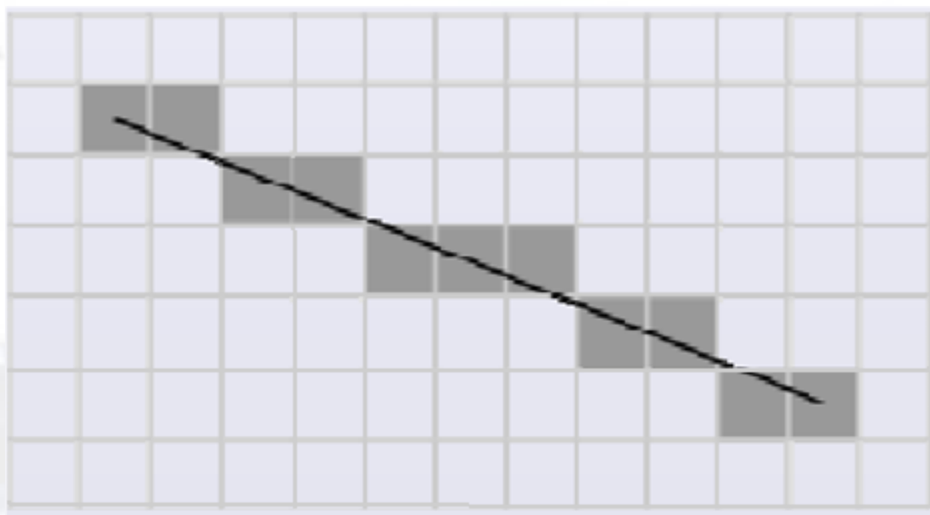
Algoritmo do Ponto-Médio ou Bresenham

- (lousa)



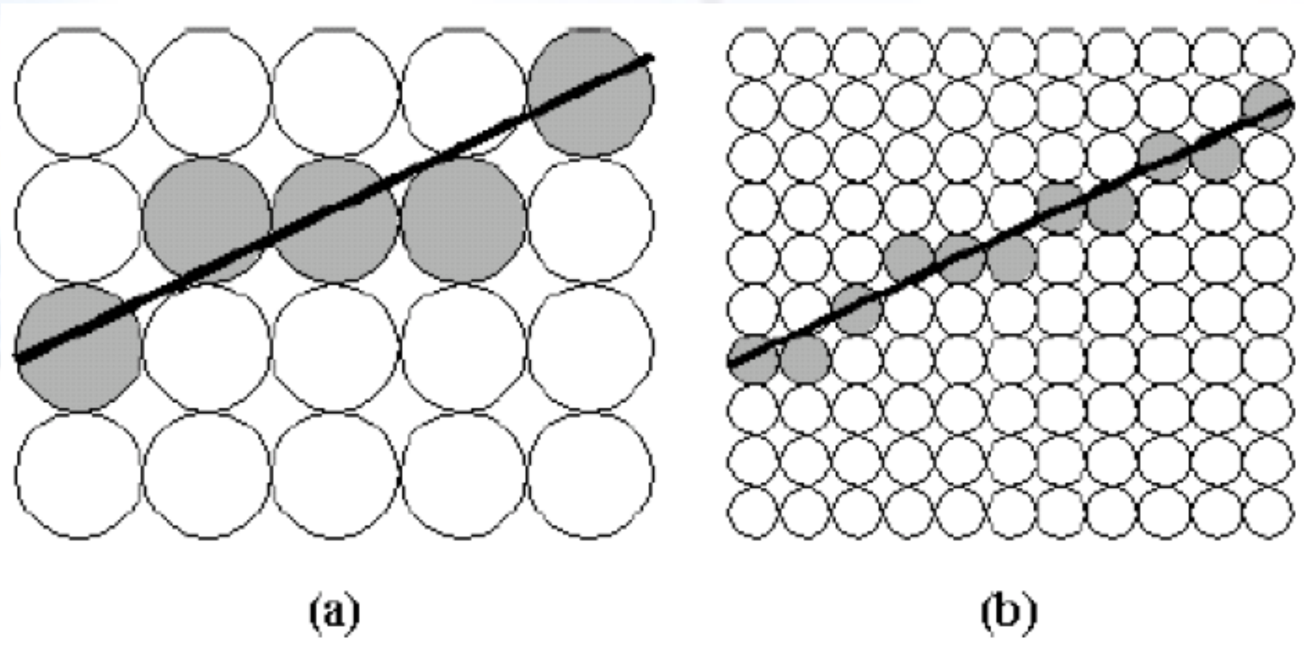
Correção no Traçado

- *Antialiasing*
  - ✓ Primitiva (ex., segmento de reta) tem espessura (área)



Correção no Traçado *(cont.)*

- *Aliasing*: efeito escada, ou serrilhado.



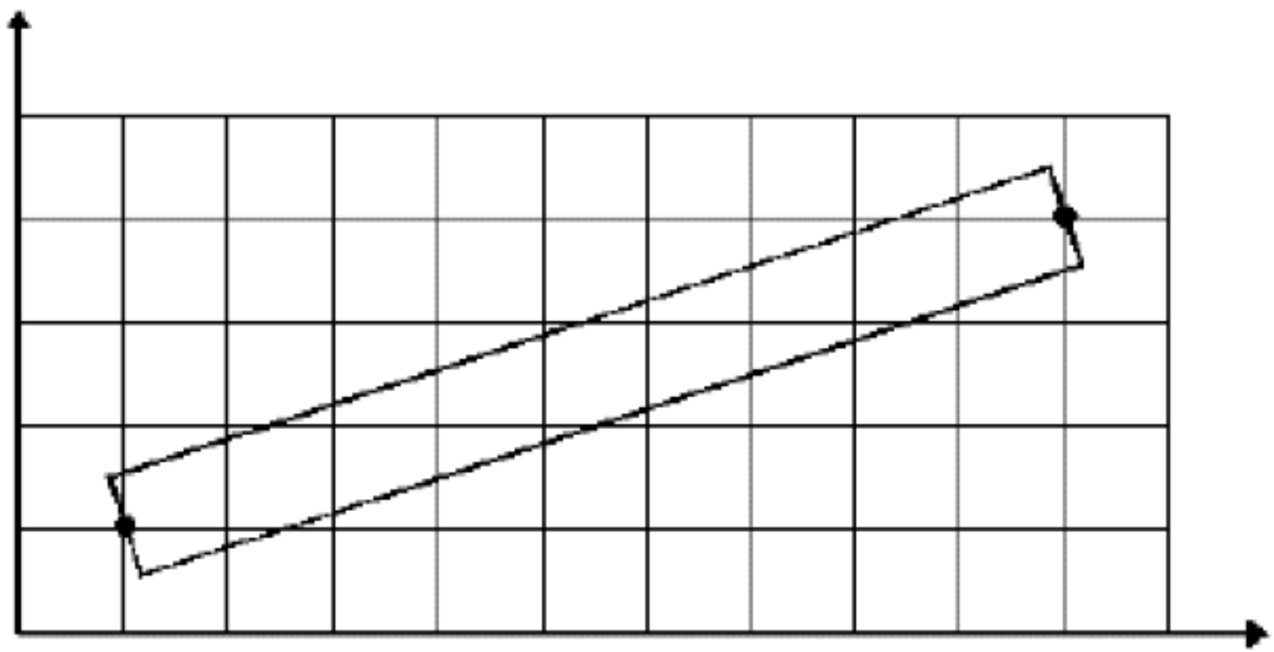
(a) é uma ampliação da região central em (b)

Correção no Traçado *(cont.)*

- *Antialiasing*: uma possível abordagem
  - Amostragem de áreas ponderada
    - Intensidade atribuída ao pixel é proporcional à sua área ocupada pela primitiva

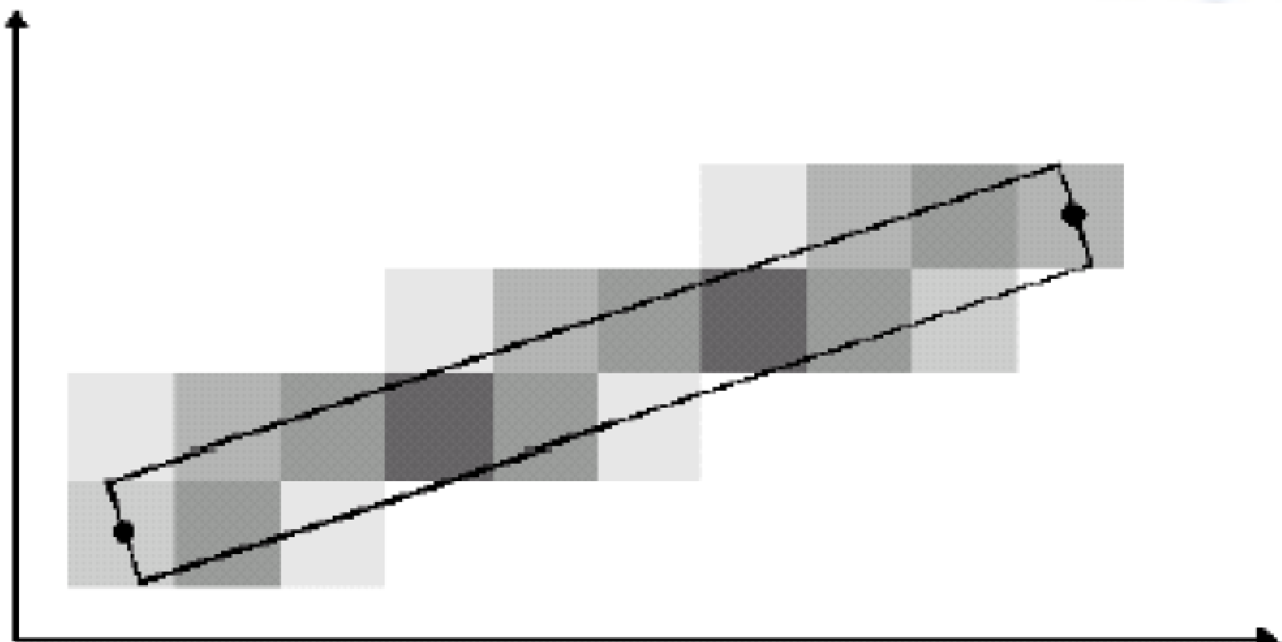
Correção no Traçado *(cont.)*

- Segmento de reta



Correção no Traçado *(cont.)*

- Intensidade do pixel proporcional à área coberta pela 'primitiva ideal'.



Referências

- Slides: Profa. Maria Cristina – ICMC/USP

- Livros:
  - ✓ Hearn, D. ; Baker, M. P. Computer Graphics, Prentice Hall
  - ✓ Foley, J.D; van Dam, A.; Feiner, S. K.; Hughes, J. F. Computer Graphics – Principles and Parctice, Addison-Wesley.