



Processamento digital de imagens

Transformações geométricas



Conteúdo

1. Operações básicas: translação, escala, rotação
2. Mapeamento direto
3. Mapeamento inverso
4. Reamostragem: Bilinear, convolução cúbica, vizinho mais próximo.
5. Transformação imagem x imagem: estimativa de parâmetros
6. aplicação



- Transformações básicas



Operações básicas

Translação

$$xs = x + t_x$$

$$ys = y + t_y$$

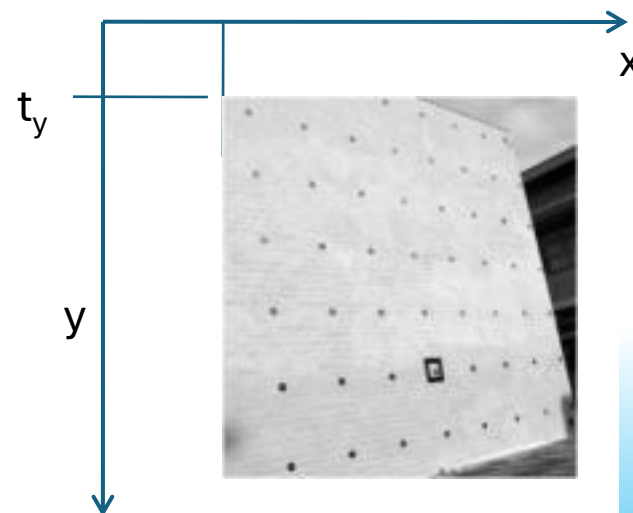
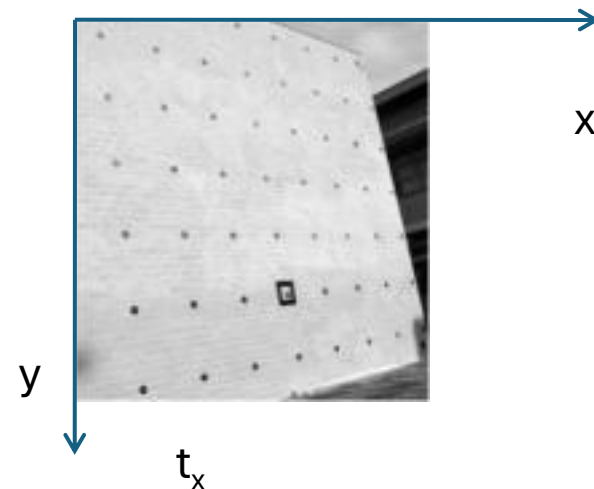
notação vetorial

$$XS = X + T \text{ (vetores)}$$

$$\begin{bmatrix} xs \\ ys \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

$$\begin{bmatrix} xs \\ ys \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} xs \\ ys \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ tx \\ 0 \\ 1 \\ ty \end{bmatrix}$$



Rotação

$$x_s = x \cos(a) - y \sin(a)$$

$$y_s = x \sin(a) + y \cos(a)$$

$$x_s = r_1 x + r_2 y$$

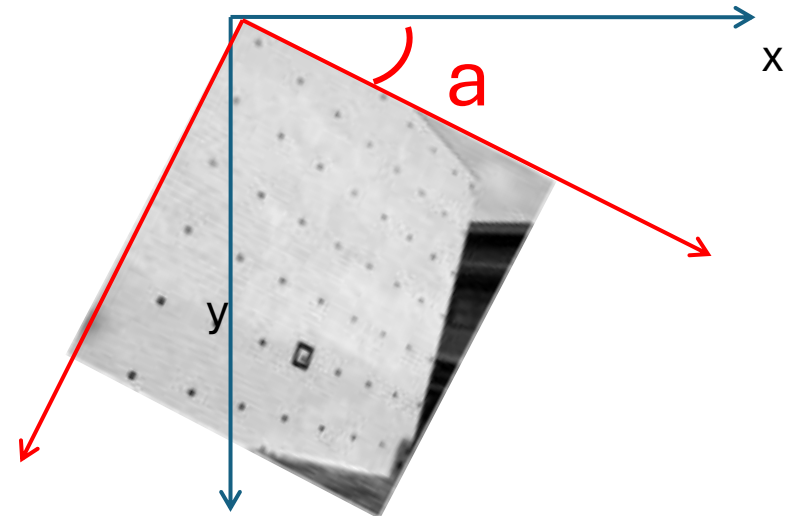
$$y_s = r_3 x + r_4 y$$

notação vetorial

$$X_s = R * X$$

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} r_1 & r_2 \\ r_3 & r_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

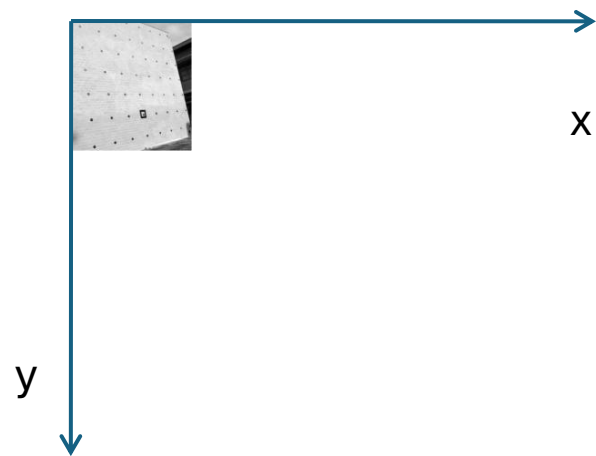
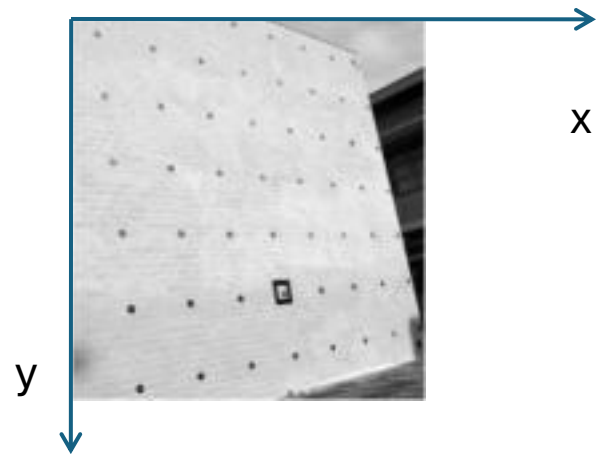
$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} * \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ a_4 \\ 0 \end{bmatrix}$$



- Escala $X_S = E * X$

$$x \begin{bmatrix} x_S \\ y_S \end{bmatrix} = E * \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x_S \\ y_S \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} * \begin{bmatrix} e_1 \\ 0 \\ 0 \\ e_2 \\ 0 \\ 0 \end{bmatrix}$$



RST

Juntando: RST (Rotation, Scale, Translation)

$$\mathbf{X}_s = \mathbf{E} * \mathbf{R} * \mathbf{X} + \mathbf{T}$$

$$\begin{bmatrix} x_s \\ y_x \end{bmatrix} = \mathbf{E} * \begin{bmatrix} r_1 & r_2 \\ r_3 & r_4 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} x_s \\ y_x \end{bmatrix} = \begin{bmatrix} Er_1 & Er_2 \\ Er_3 & Er_4 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} x_s \\ y_x \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_3 \\ a_6 \end{bmatrix}$$

$$\begin{bmatrix} x_s \\ y_x \end{bmatrix} = \begin{bmatrix} x & a_1 & y & a_2 & a_3 \\ x & a_4 & y & a_5 & a_6 \\ 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_s \\ y_x \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix}$$

Transformação espacial

Se conhecemos os parâmetros da transformação

por ex: $\mathbf{X}_s = \mathbf{E} * \mathbf{R} * \mathbf{X} + \mathbf{T}$

Podemos aplicar a transformação espacial para calcular a posição de um pixel na imagem de saída.

- Ou seja, dadas as coordenadas na imagem original (x,y) calculamos as coordenadas na imagem de saída $u=f(x,y)$, $v=f(x,y)$ e copiamos o valor digital nessa posição da imagem nova.



1. Mapeamento direto



Mapeamento direto

Dadas as coordenadas da imagem de entrada (x,y), calcular a nova posição na imagem e saída (u,v) e copiar o valor digital.

Ex:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_3 \\ a_6 \end{bmatrix}$$

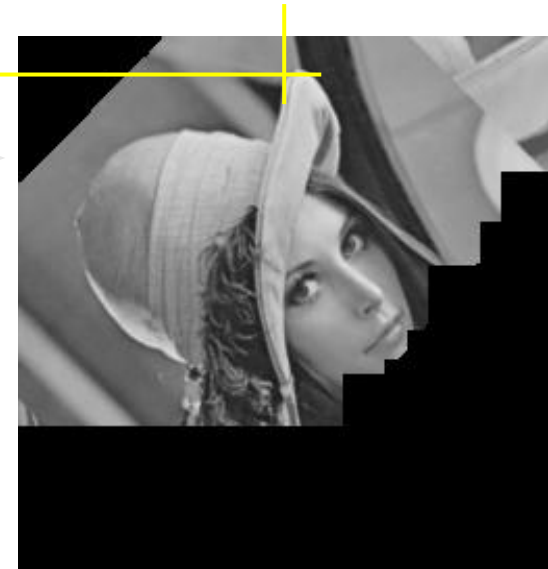


x,y



$u,v=f(x,y)$

Arredondar (u,v)



u,v

código

Dado que conhece a imagem de entrada (E), os parâmetros de transformação (a), e criou uma imagem em branco para a saída...

#Varrer a imagem de ENTRADA (E) para produzir saída (S)

For lin in range(Nlin)

 For col in range(Ncol)

calcule lin0/col0 da imagem de saída

$col0 = a0 * col + a1 * lin + a3$

$lin0 = a4 * col + a5 * lin + a6$

 if $col0 > 0$ & $col0 < Ncol$ & $lin0 > 0$ & $lin0 < Nlin$

verificar se está dentro da imagem de saída

copiar valor

$S(lin, col) = E(lin, col)$

tarefa

- Leia uma imagem quadrada (E)
- Calcule os parâmetros de transformação para efetuar a rotação da imagem em 45 graus em relação ao centro da imagem.
- Mostre o resultado.



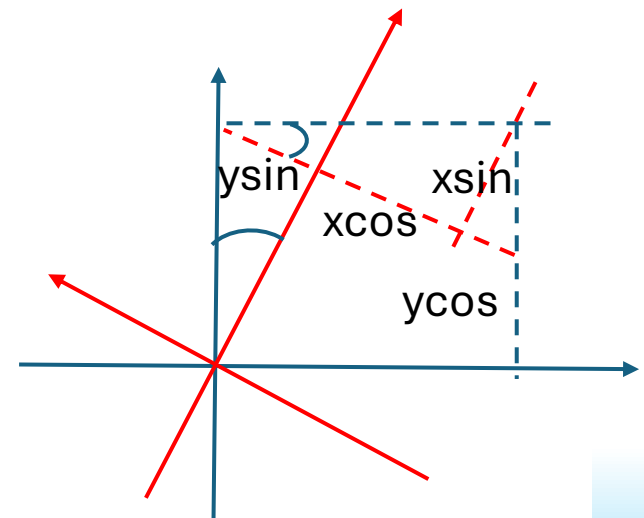
A rotação 2D pode ser escrita como:

- $\begin{bmatrix} u \\ v \end{bmatrix} = R * \begin{bmatrix} x \\ y \end{bmatrix}$

com

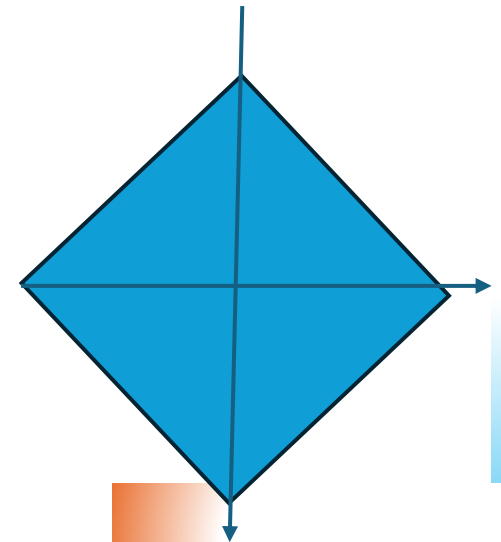
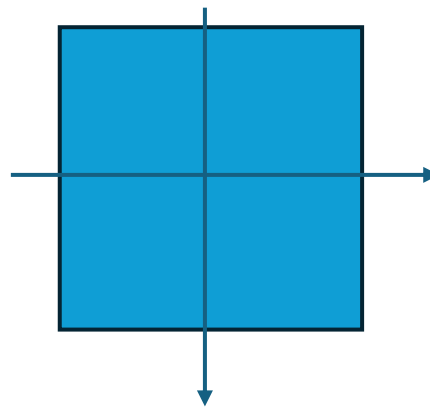
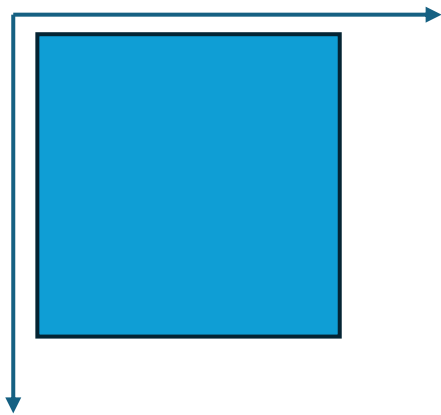
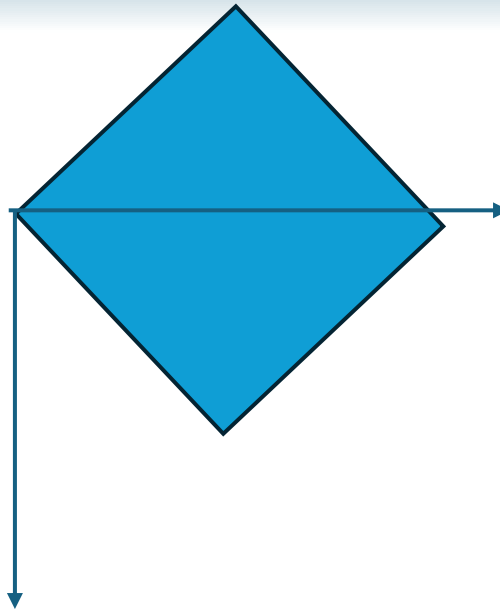
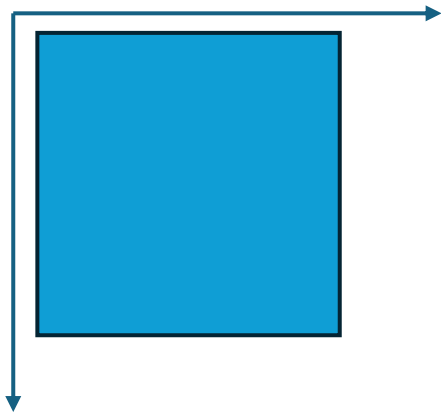
- $R = \begin{bmatrix} \cos(ang) & -\sin(ang) \\ \sin(ang) & \cos(ang) \end{bmatrix}$

Obs. Ângulo em radianos



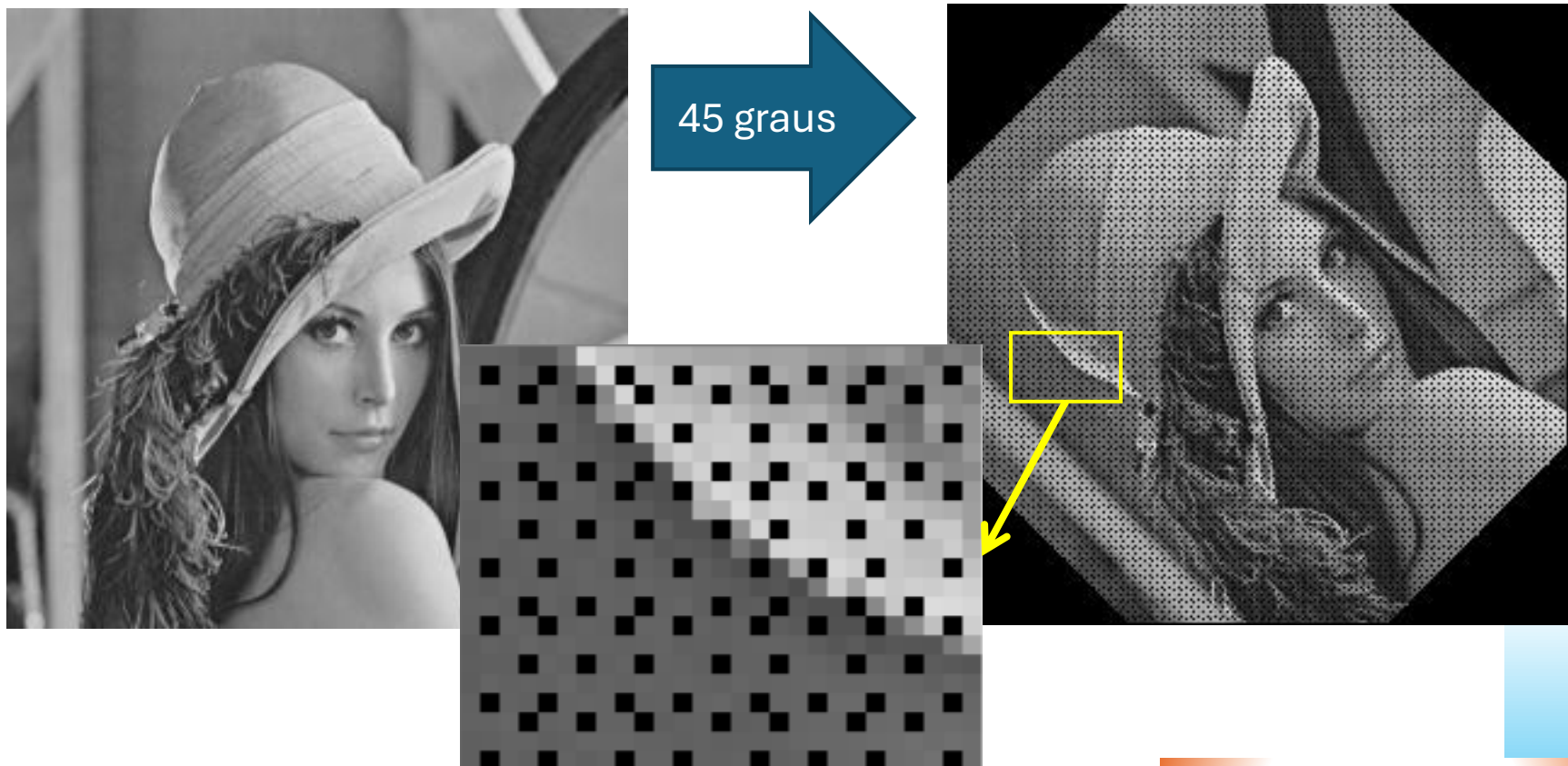
dicas

- considere



Problemas

Nem todas as posições da imagem de saída são ocupadas devido a arredondamentos.





Mapeamento inverso



Mapeamento inverso

Dadas as coordenadas da imagem de saída, calcular a posição na imagem de entrada e copiar o valor digital.

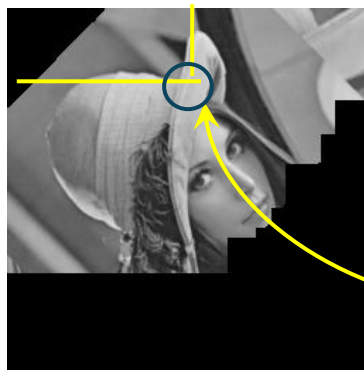
Ex RST: Se temos a transformação

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_3 \\ a_6 \end{bmatrix}$$

Podemos calcular a transformação inversa

$$\begin{bmatrix} u - a_3 \\ v - a_6 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \text{inv} \left(\begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} \right) * \begin{bmatrix} u - a_3 \\ v - a_6 \end{bmatrix}$$



u,v



$x,y=f(u,v)$

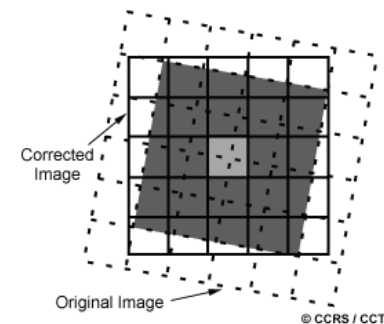
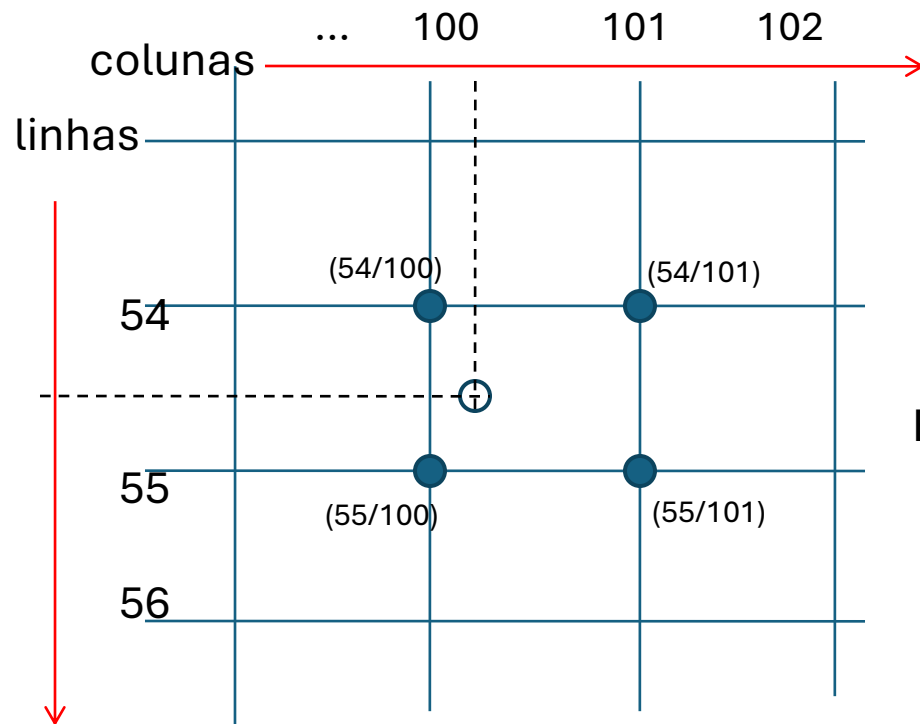
Valor digital



x,y

interpolação

As coordenadas calculadas nem sempre correspondem a números inteiros e por este motivo o novo valor digital deve ser interpolado. Existem para isto três opções mais conhecidas que são a reamostragem pelo método do vizinho mais próximo, a interpolação bilinear e a convolução cúbica.

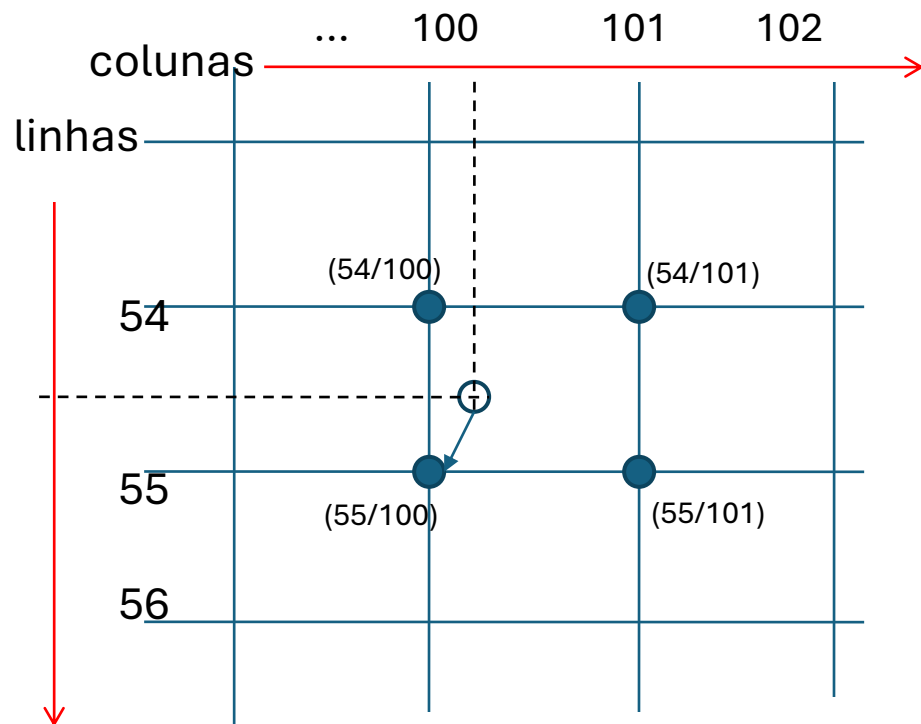
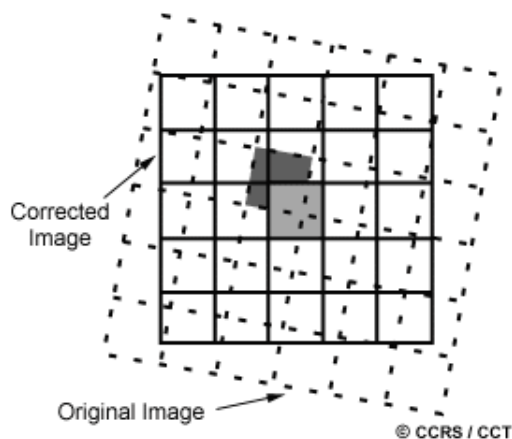


Ex: após o cálculo obtivemos:
linha=54,6
coluna= 100,3
Qual valor copiamos?

vizinho mais próximo

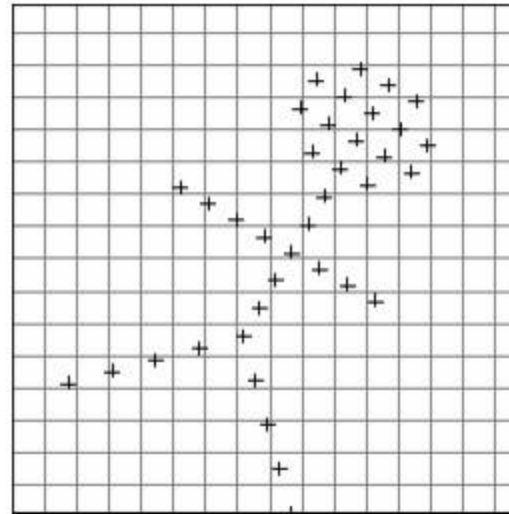
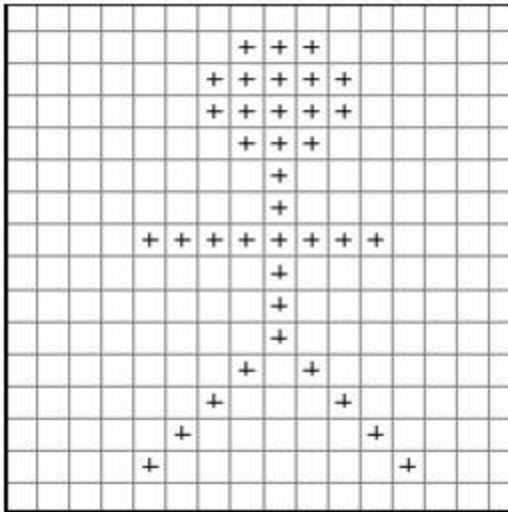
É mais simples e consiste na escolha do valor do contador digital do pixel mais próximo.


Como um valor é copiado, não gera novos valores interpolados.



Equivale a arredondar os valores em linha e coluna para o inteiro mais próximo
(54,6; 100,3) ... (55,100)

Exemplo:





Dado que conhece a imagem de entrada (E), os parâmetros da transformação inversa *a) e alocou memória para uma saída (S)

Varrer a imagem de saída

For lin in range(Nlin)

For col in range(Ncol)

calcule lin0/col0 da imagem original

$col0 = a0 * col + a1 * lin + a3$

$lin0 = a4 * col + a5 * lin + a6$

if col0>0 & col0<Ncol & lin0>0 & lin0<Nlin

verifique se está dentro da imagem de entrada

#interpolador valor, ainda por fazer...

interpolado= interpola(E,lin,col)

interpolado=np.uint8(interpolado)

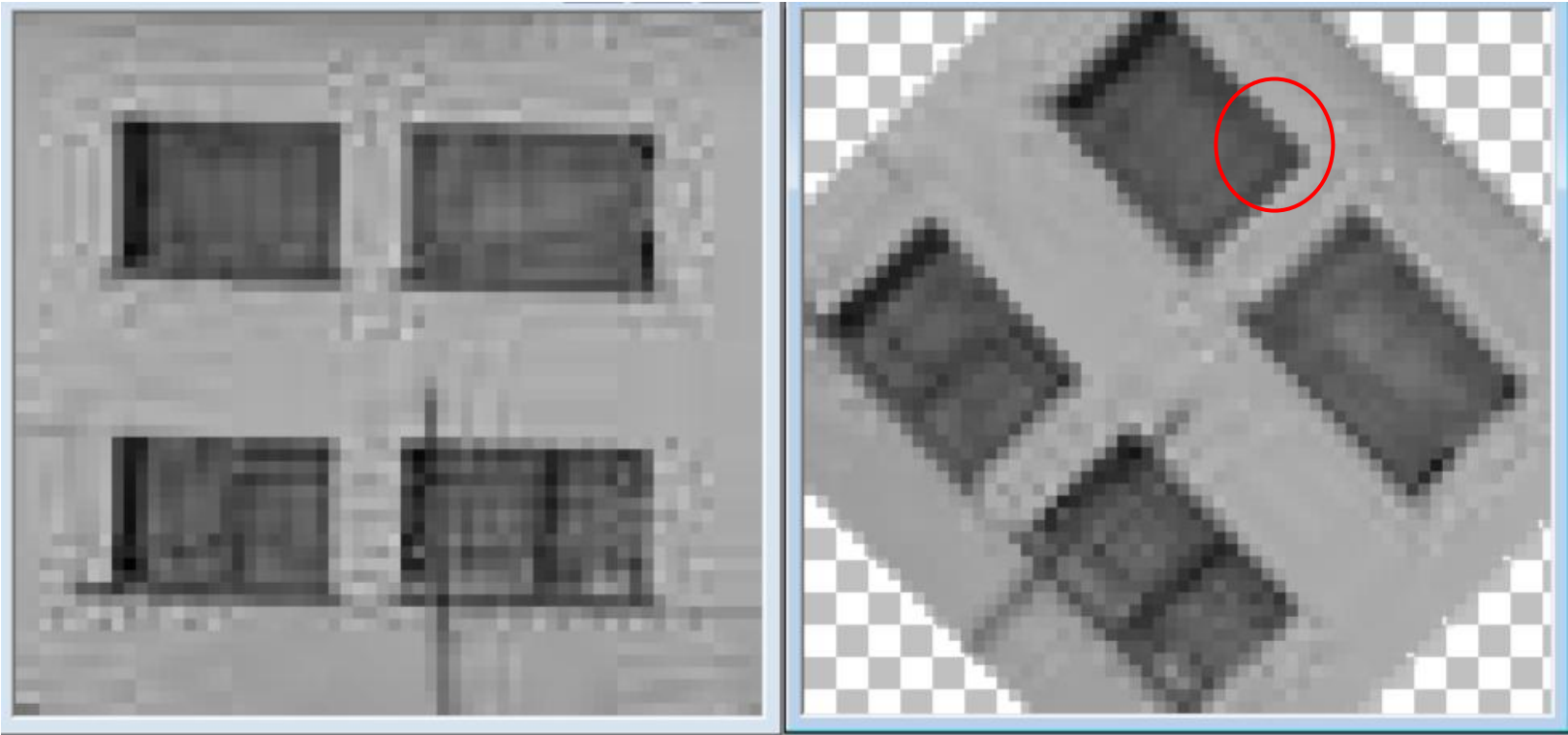
copiar o valor no pixel na matriz de saída

J(lin, col)=interpolado



Efeito da interpolação (suavização)

- Produz um efeito de degrau em imagens de nível de cinza, devido ao arredondamento da posição do pixel na imagem original.



Interpolação

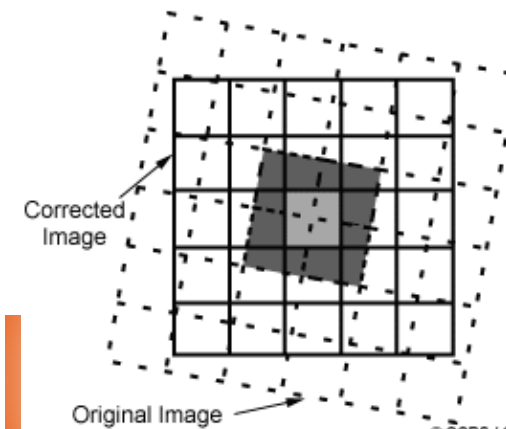
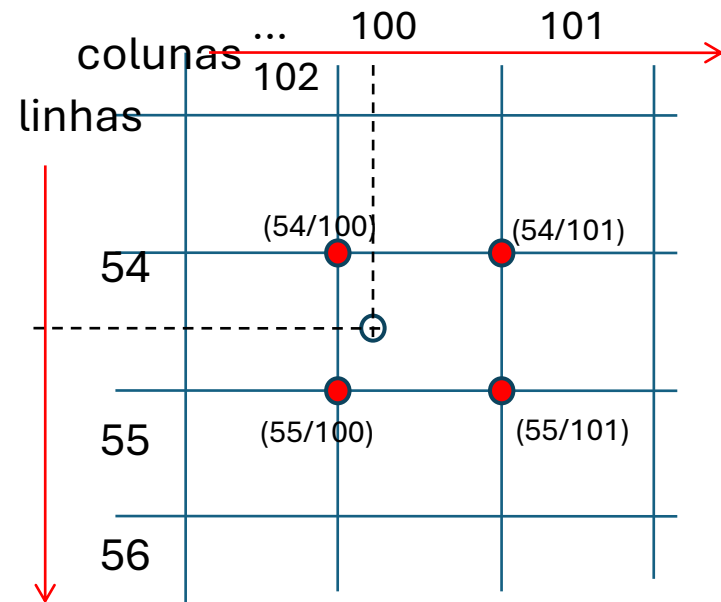
Reamostragem: Bilinear, convolução cúbica, vizinho mais próximo.

- Vizinho mais próximo
- Bilinear
- Convolução cúbica

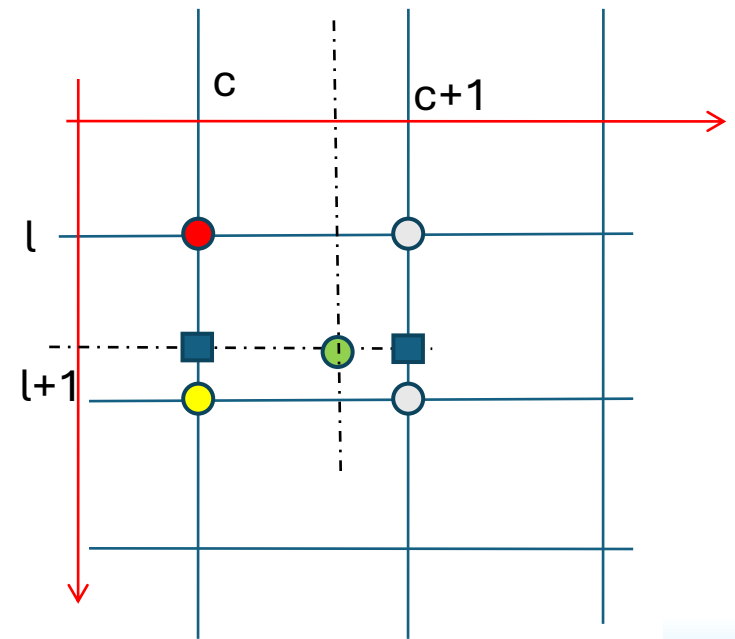
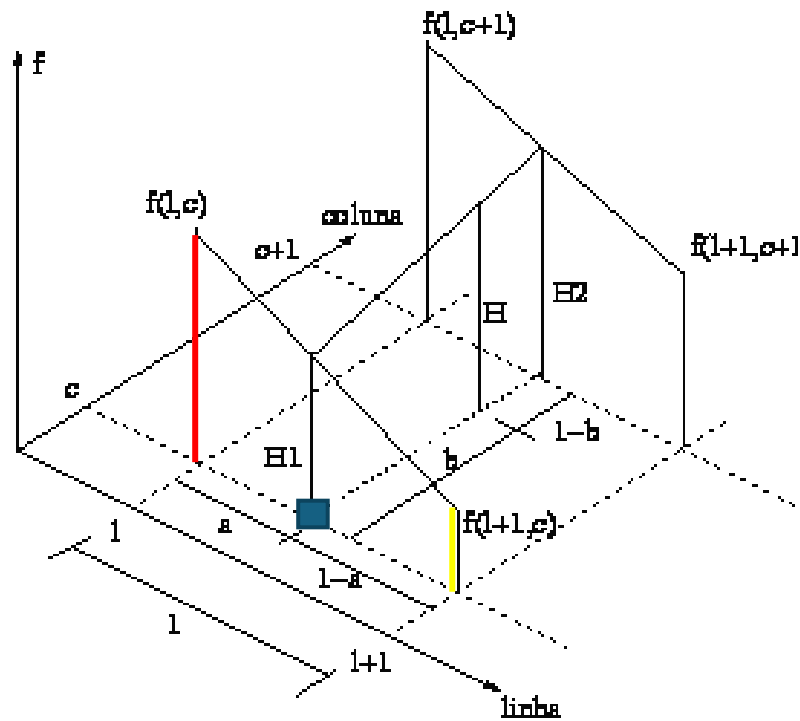
Interpolação bilinear: Consiste em interpolar um novo valor a partir dos quatro vizinhos mais próximos (linha e coluna anterior e posterior).

Poderíamos usar o valor médio dos quatro vizinhos, mas isso criaria áreas uniformes quando se muda a escala. Por isso, se interpola com variação dentro desta vizinhança.

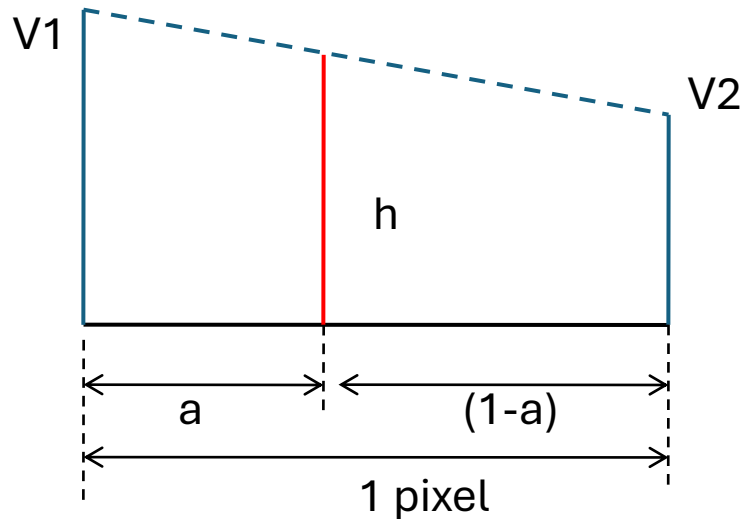
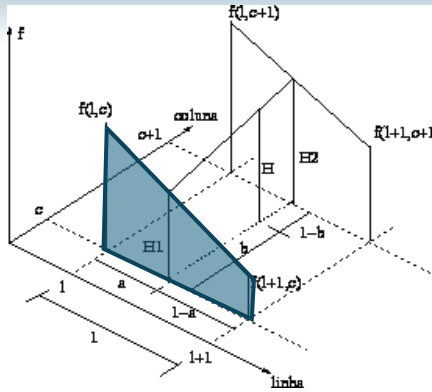
Para isto se faz interpolações em linhas e em colunas.



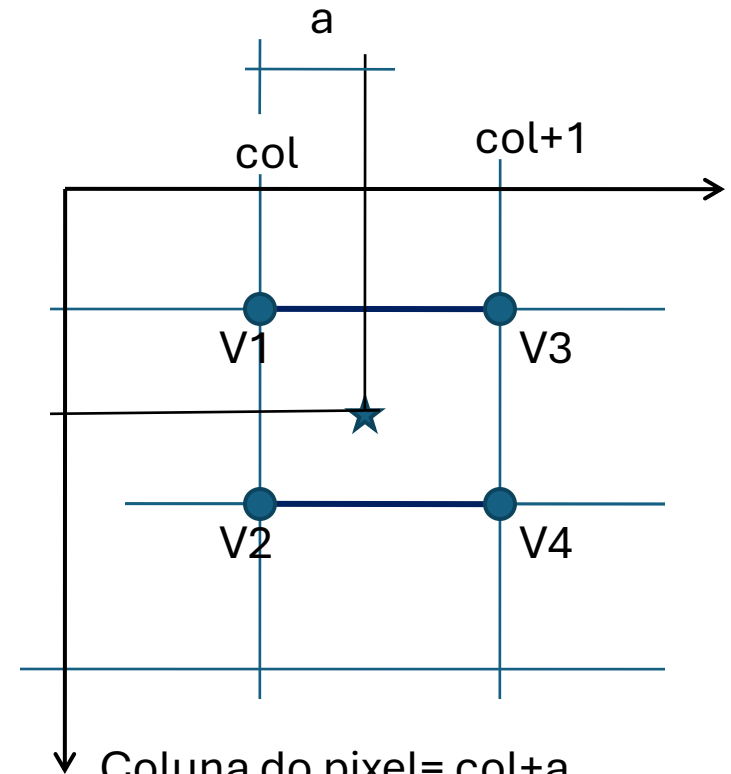
- Interpolação Bilinear



Ao longo de uma linha



$$h = V1 * (1-a) + v2 * a$$



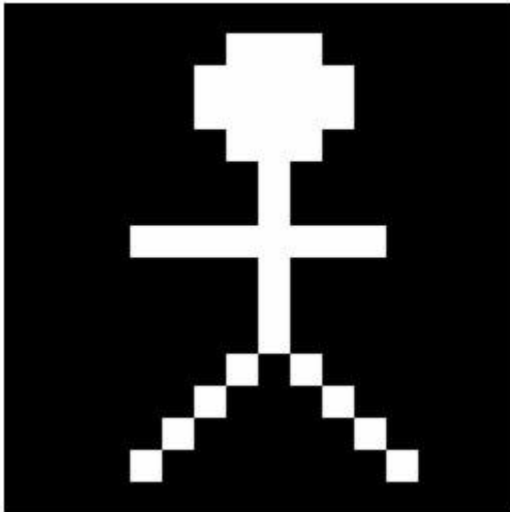
Coluna do pixel = col + a

Ex: 321,81

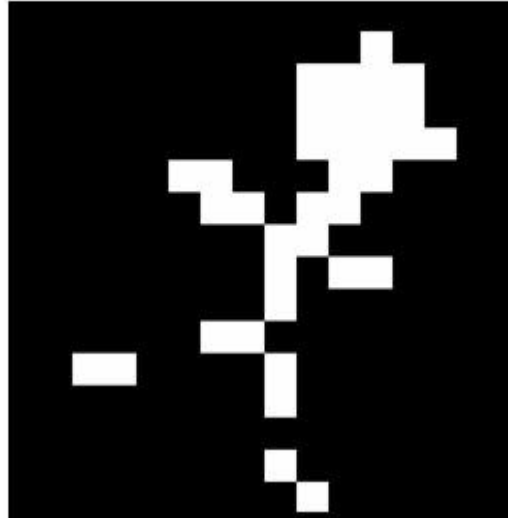
col = 321 a = 0,81

comparação

Original Image



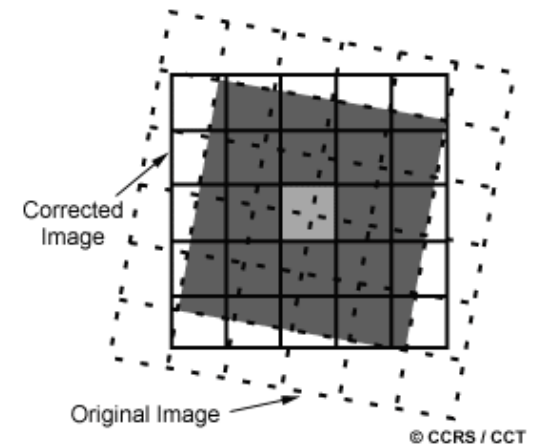
Nearest Neighbor



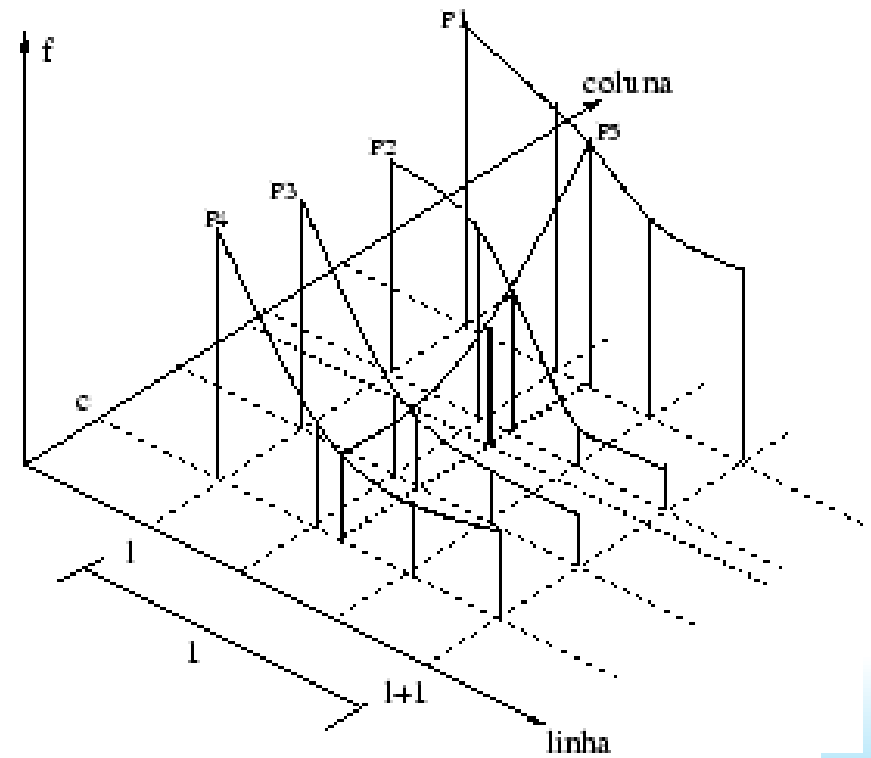
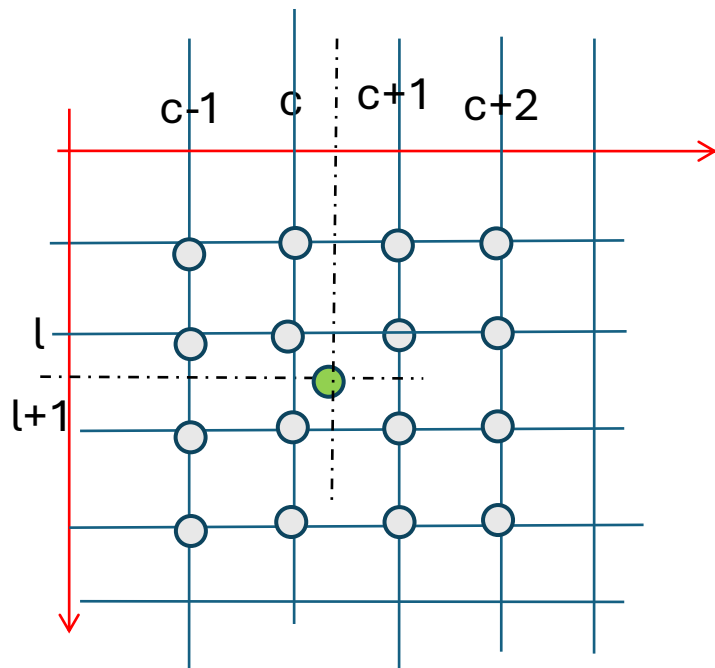
Bilinear Interpolation



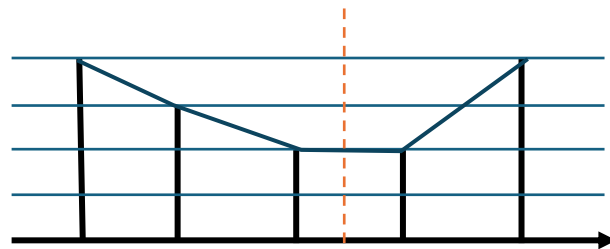
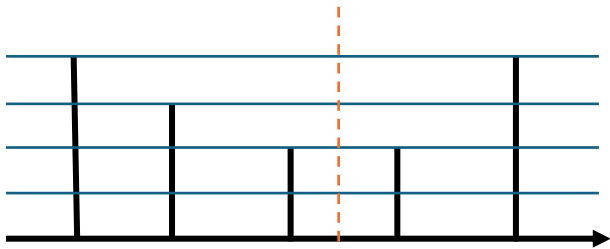
- Convolução cúbica: Consiste em interpolar um novo valor a partir dos 16 vizinhos mais próximos,
- utilizando funções cúbicas para a interpolação.



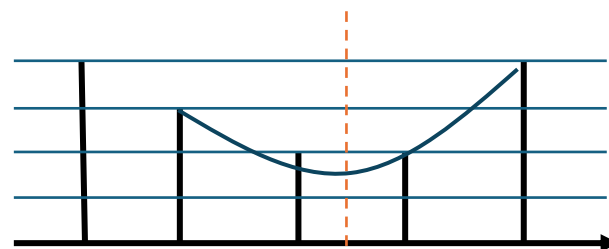
Convolução cúbica. Usa funções cúbicas



- O que ocorreria se usarmos diferentes interpoladores nesta linha?

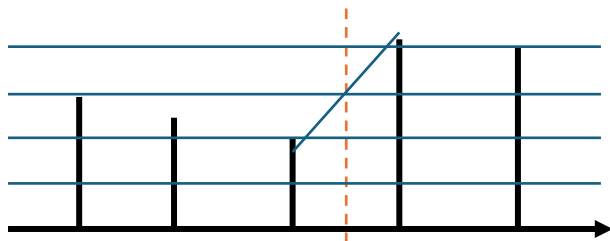
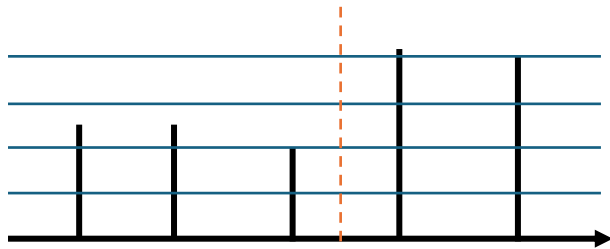


linear

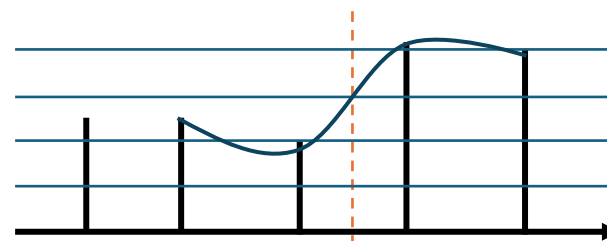


cúbico

- O que ocorreria se usarmos diferentes interpoladores nesta linha?



linear



cúbico



- Transformação imagem x imagem: estimativa de parâmetros
- Aplicação



Estimativa de Modelo de Transformação

imagem

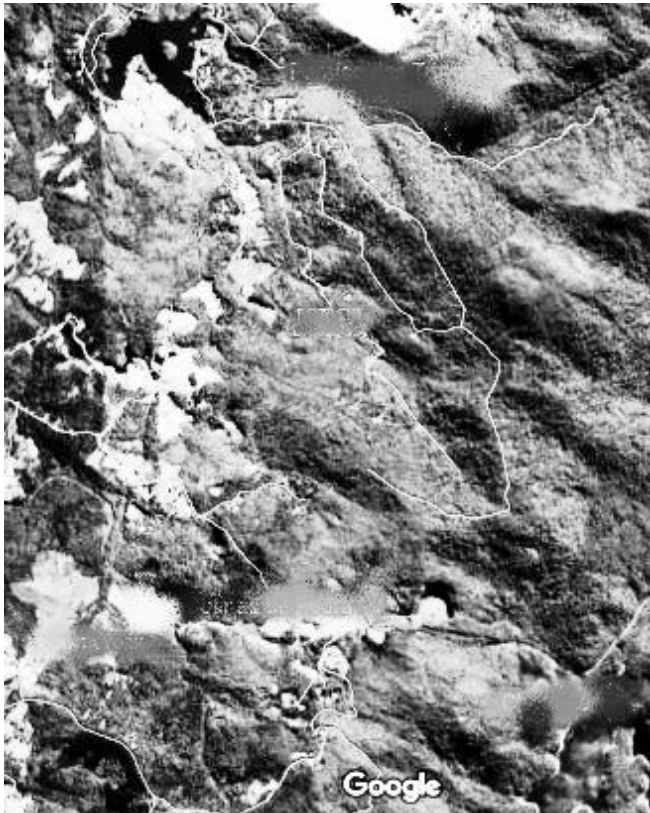


Imagem 2



Finalidade:

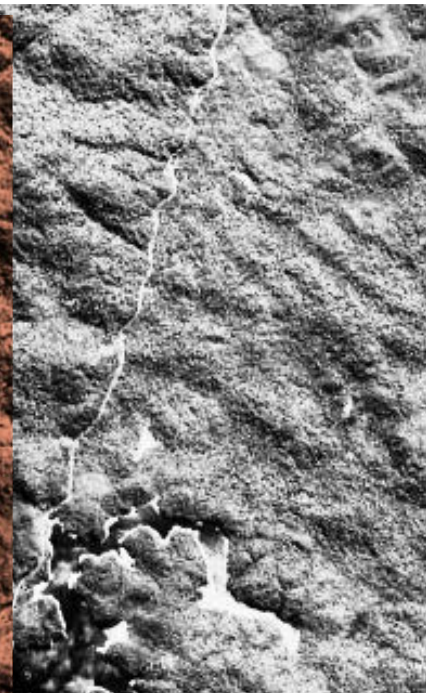
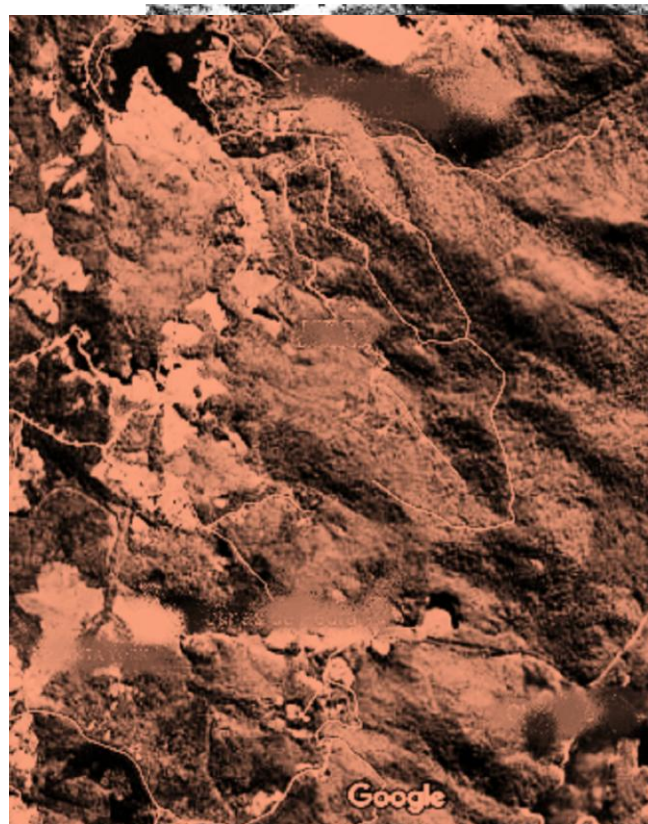
Transformar a geometria da primeira imagem e ajustá-la à geometria da segunda (base/referencia)

Estimativa de Modelo de Transformação

imagem

Imagem 2

Finalidade:
Transformar a
geometria da
primeira imagem e
ajustá-la à
geometria da
segunda
(base/referencia)



Pontos homólogos

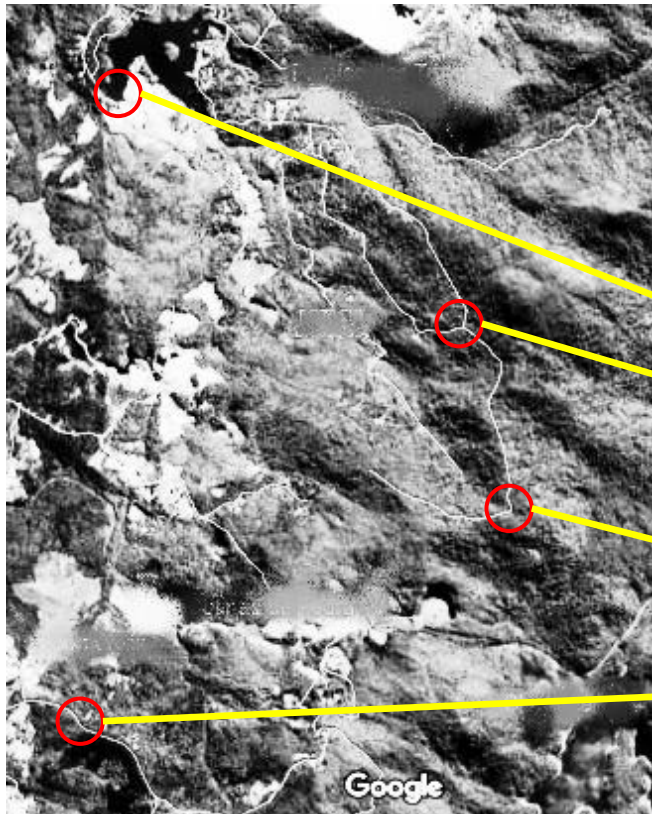


Imagem w

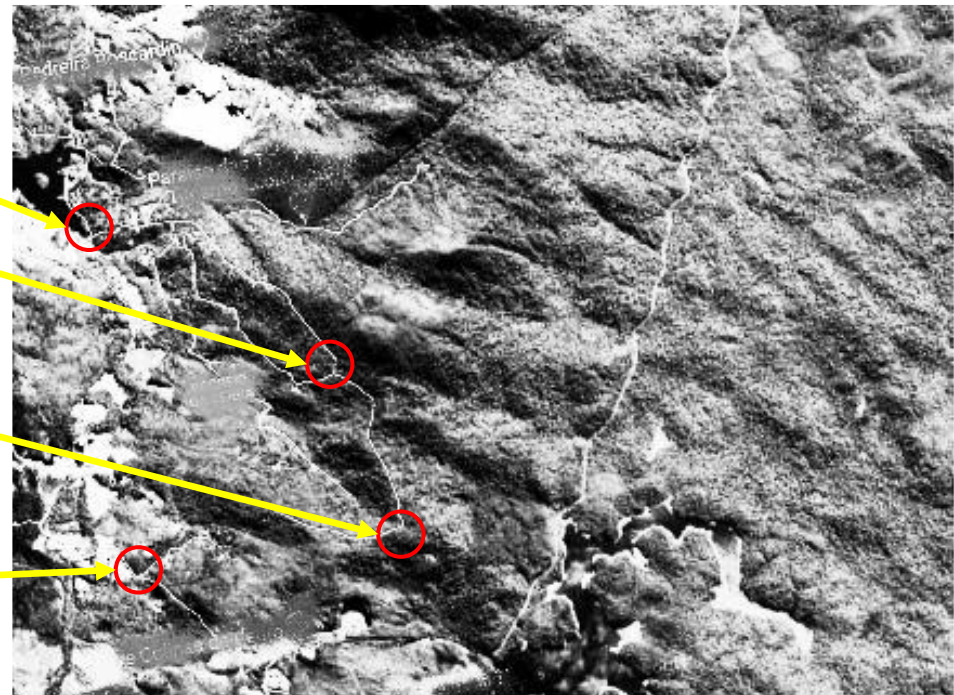


Imagem 2

Coordenadas dos pontos homólogos

	Imagem 1		Imagem 2	
ponto	col	lin	col	lin
	1x1	y1	u1	v1
	2x2	y2	u2	v2
	3x3	y3	u3	v3
...
n	xn	yn	vn	vn

Após coletar pontos na imagem a ser modificada e a que servirá de base para a transformação, espera-se obter uma transformação que permita transformar (x,y) para (u,v) ou vice-versa.

$$(x,y) = F(u,v)$$

Ou

$$(u,v) = F(x,y)$$

Estimativa de Modelo de Transformação

Transformação polinomial de primeiro grau:

$$\begin{bmatrix} x_s \\ y_x \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_3 \\ a_6 \end{bmatrix}$$

Permite calcular as coordenadas x, y a partir da combinação (linear) das coordenadas u, v .

Para isto, é necessário conhecer seis parâmetros, os quais podem ser obtidos matematicamente.

Reescrevendo:

$$\begin{bmatrix} x_s \\ y_x \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix}$$

Para 3 pontos

Com um ponto obtemos 2 equações, porém usando “n” pontos podemos ter $2 \times n$ equações

$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2 & v_2 & 1 \\ u_3 & v_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_3 & v_3 & 1 \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix}$$

Para “n” pontos

Com um ponto obtemos 2 equações, porém usando “n” pontos podemos ter 2xn equações

$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ x_4 \\ \dots \\ x_n \\ y_n \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2 & v_2 & 1 \\ u_3 & v_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_3 & v_3 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ u_n & v_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_n & v_n & 1 \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} \rightarrow S = E * A$$

$$S(2n \times 1) = E(2n \times 6) * A(6 \times 1)$$

solução

$$S = E * A \quad [\text{vetor}(2n \times 1) = \text{matriz}(2n * 6) * \text{vetor}(6 \times 1)]$$

Queremos calcular os parâmetros contidos no vetor “A”

$$S = E * A$$

$$\text{Transposta}(E) * S = \text{Transposta}(E) * E * A$$

$$E' * S = E' * E * A$$

$$E' * S = (E' * E) * A$$

$$\text{Inversa}(E' * E) * E' * S = \text{Inversa}(E' * E) * (E' * E) * A$$

Como $\text{inv}(M) * M = \text{matriz identidade}$:

$$A = \text{inv}(E' * E) * E' * S$$

Logo, podemos calcular as coordenadas da transformação desde que tenhamos pelo menos 3 pontos. Na prática usamos mais pontos (use pelo menos 5)

Trabalho

- Usando duas fotografias aéreas com superposição, um par estéreo, determine as coordenadas de cinco pontos homólogos nas imagens.
- Calcule os parâmetros da transformação polinomial de primeiro grau.
- Verifique seu resultado. Aplicando a transformação, deveria ser possível calcular as coordenadas lidas na tela por você (x,y) , usando como entrada as coordenadas (u,v) da tabela.
- Aplique a transformação à imagem e compatibilize a geometria das duas imagens.
- Para a reamostragem, use a reamostragem bilinear.

Prática

- Utilizando uma fotografia preto e branco, aplique uma rotação de 30 graus à imagem, em relação ao centro da imagem.
- A) mapeamento direto
- B) mapeamento inverso: bilinear



interpolação bilinear

obtenha a linha/coluna anterior por arredondamento para baixo

```
L=math.floor(lin0)
```

```
C=math.floor(col0)
```

```
# calcule o resíduo em L e C
```

```
DL=lin0-L
```

```
DC=col0-C
```

```
H1=I(Dl,Dc) *(1-DL) + I(Dl+1,Dc) *DL
```

```
H2=I(Dl,Dc+1)*(1-DL) + I(Dl+1,Dc+1) *DL
```

```
H=H1*(1-DC) + H2*DC
```

