Processamento digital de imagens

Geometria

Operações básicas

Translação

$xs=x+t_x$ $ys=y+t_y$

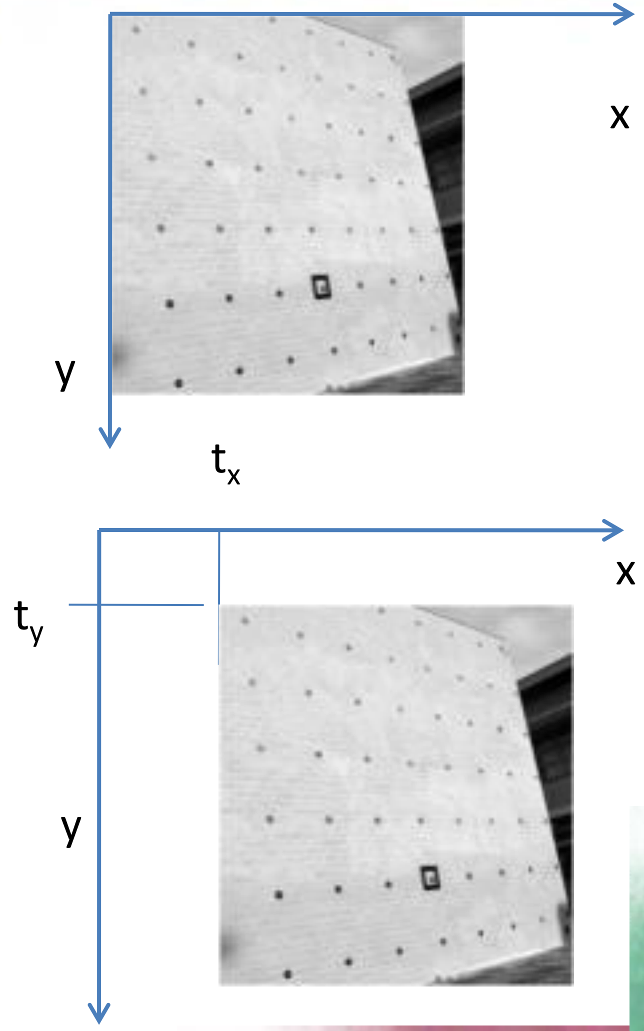
notação vetorial

$$XS = X + T \ (vetores)$$

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

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

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



Rotação

$$xs = x\cos(a) - y\sin(a)$$
$$ys = x\sin(a) + y\cos(a)$$

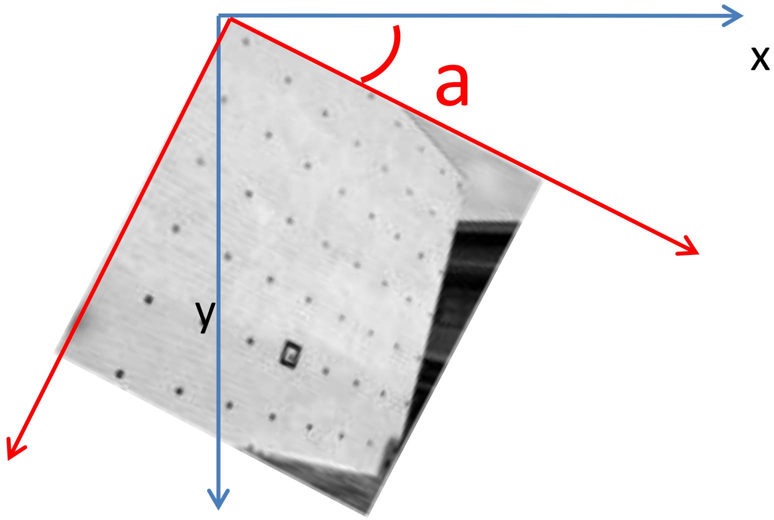
$$xs = r1\,x + r2\,y$$
$$ys = r3\,x + r4\,y$$

notação vetorial

*XS = R * X*

$$\begin{bmatrix} xs \\ y2 \end{bmatrix} = \begin{bmatrix} r1 & r2 \\ r3 & r4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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



- Escala `XS = E * X`

$$x\begin{bmatrix} xs \\ y2 \end{bmatrix} = E * \begin{bmatrix} x \\ y \end{bmatrix}$$

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

RST

Juntando: RST (Rotation, Scale,Translation)

Xs= E * R ***X** +T

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

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

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

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

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

Transformação espacial

Se conhecemos os parâmetros da transformação

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

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

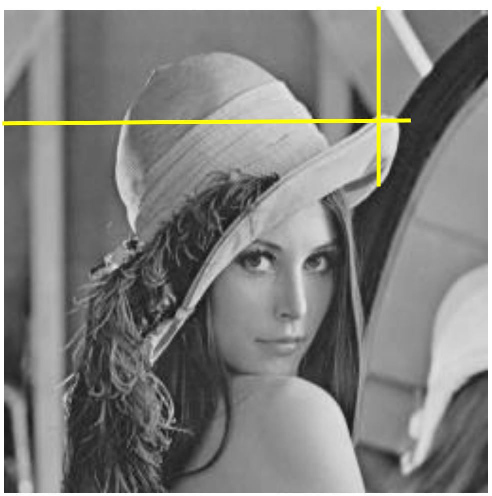
- Ou seja, dadas as coordenadas na imagem original (x,y) calculamos as coordenadas na imagem de saída u=f(x,y), v=f(x,y) e copiamos o valor digital nessa posição da imagem nova.

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:

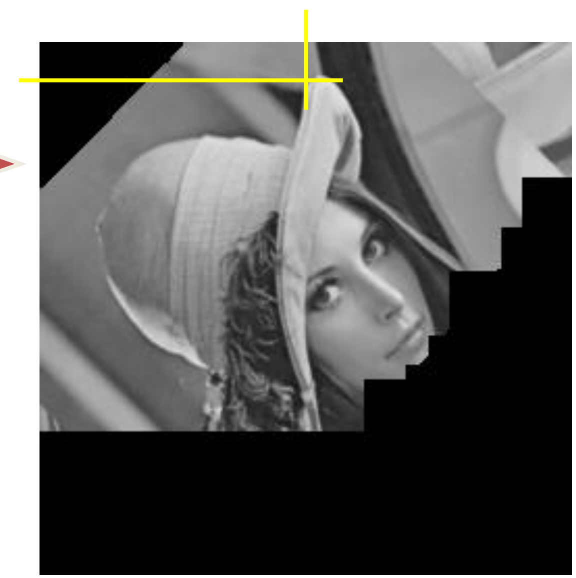
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_3 \\ a_6 \end{bmatrix}$$



x,y

u,v=f(x,y)

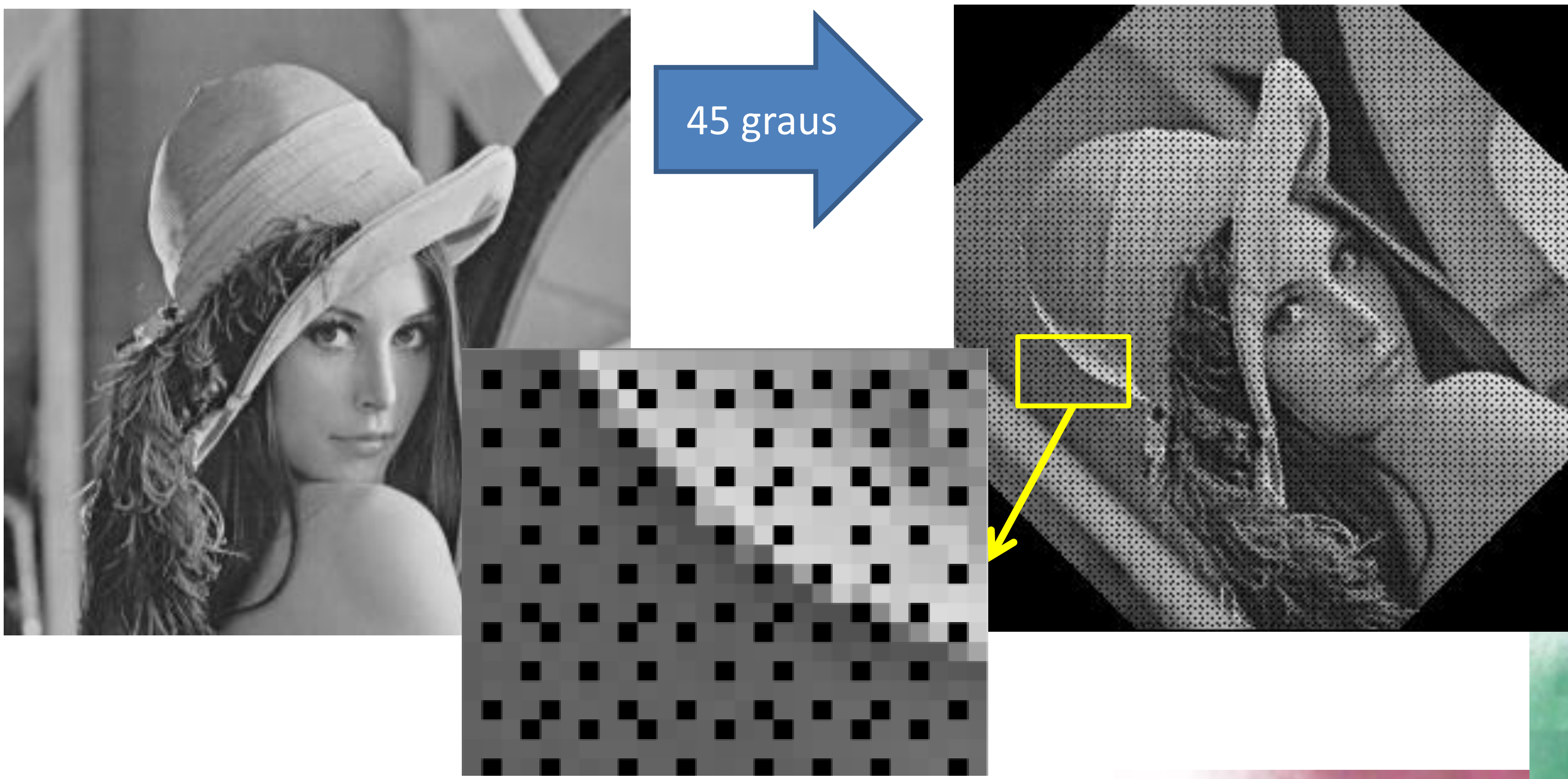
Arredondar (u,v)



u,v

Problemas

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



Mapeamento inverso

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

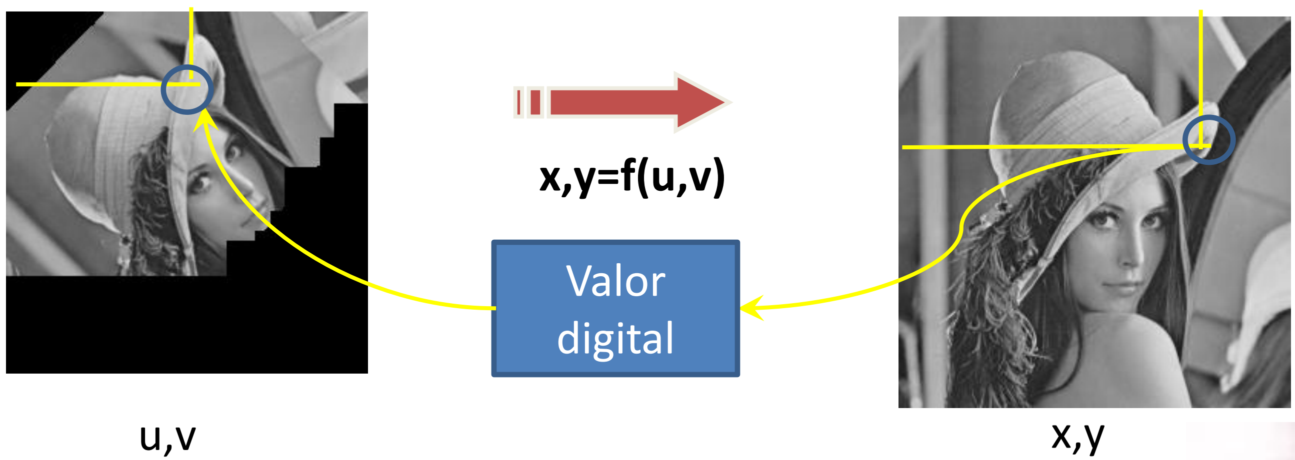
Ex RST: Se temos a transformação

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

Podemos calcular a transformação inversa

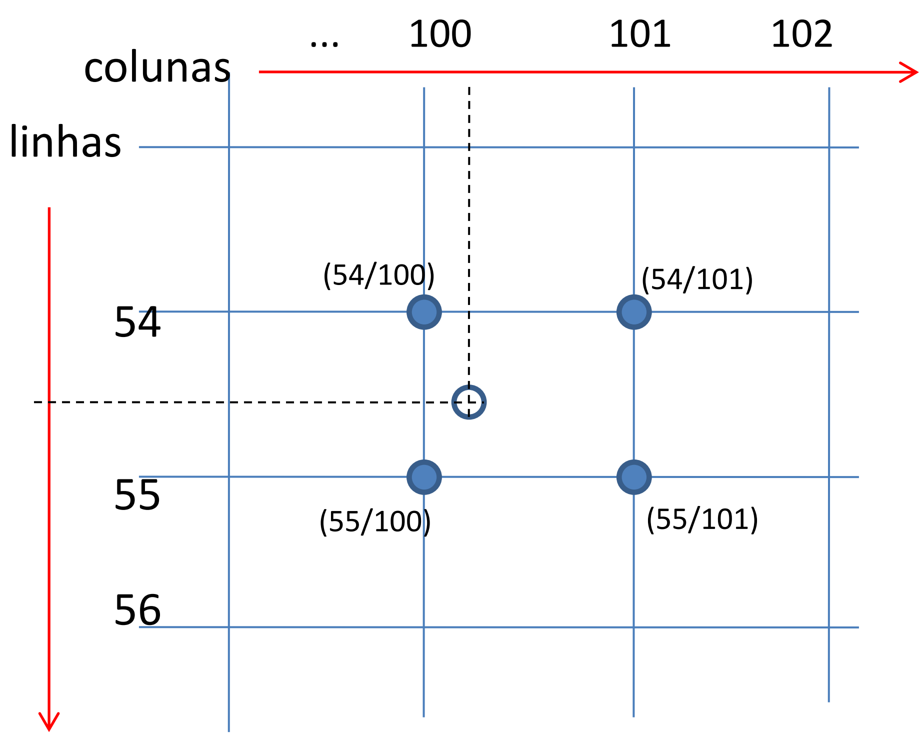
$$\begin{bmatrix} u - a_3 \\ v - a_6 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

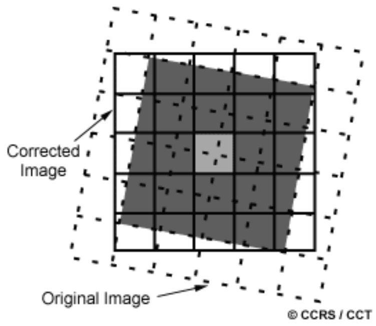
$$\begin{bmatrix} x \\ y \end{bmatrix} = inv\left(\begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix}\right) * \begin{bmatrix} u - a_3 \\ v - a_6 \end{bmatrix}$$



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.



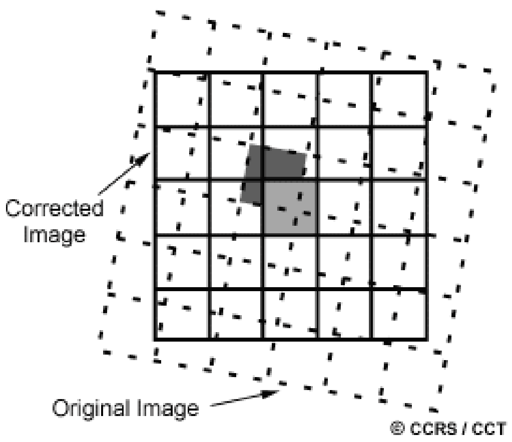


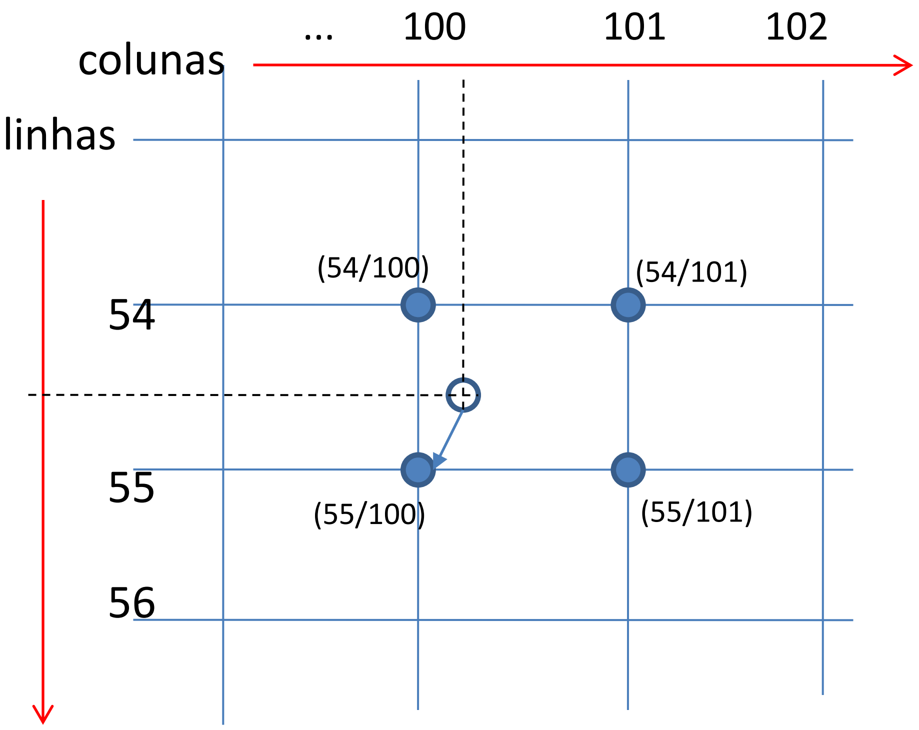
Ex: após o cálculo obtivemos:
linha=54,6
coluna= 100,3
Qual valor copiamos?

vizinho mais próximo

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

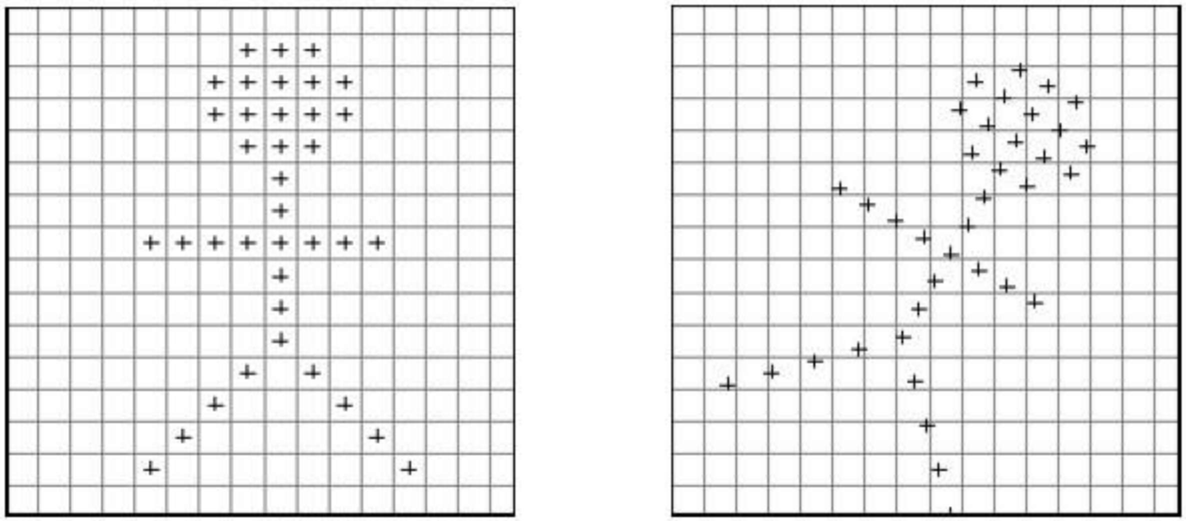
Como um valor é copiado, não gera novos valores interpolados.





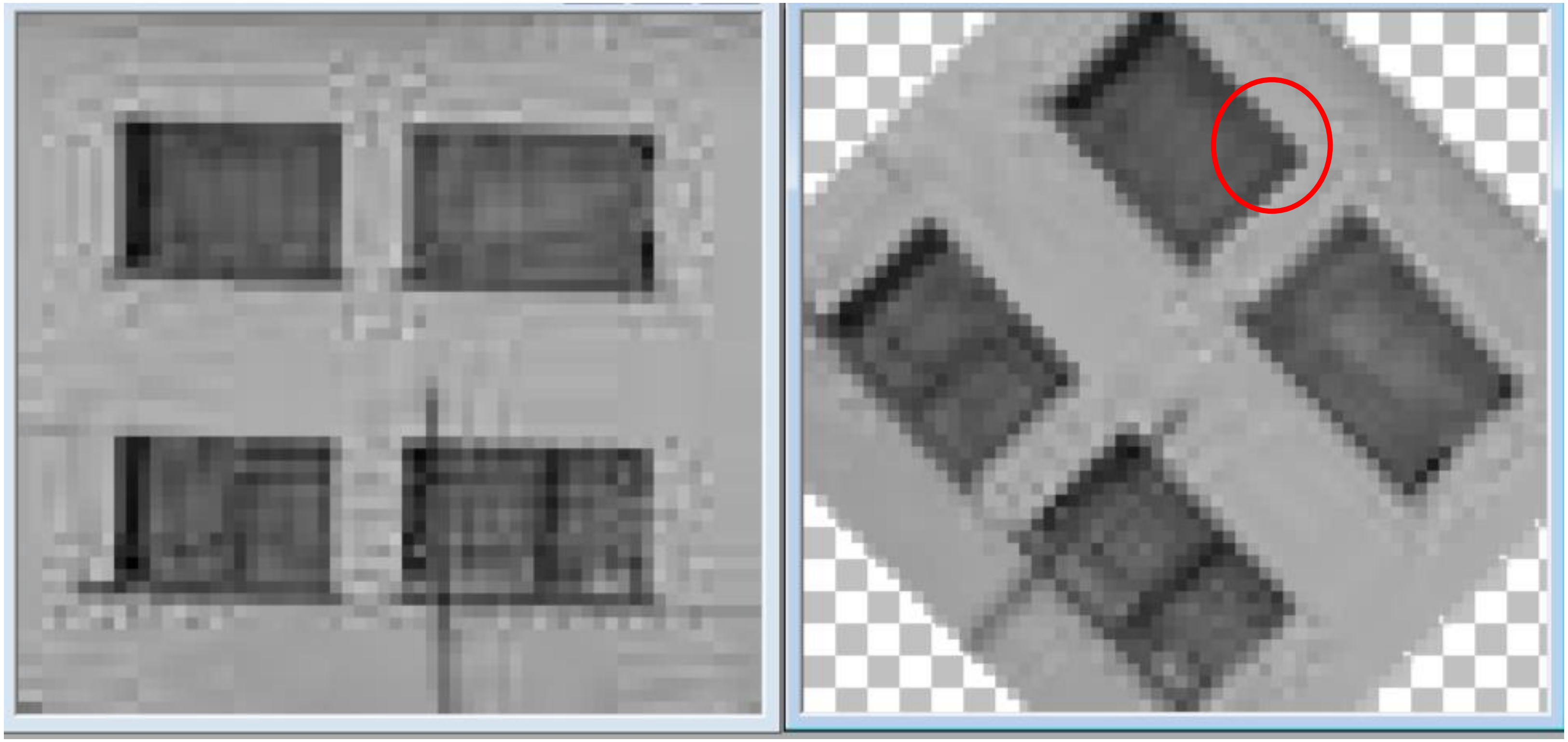
Equivale a arredondar os valores em linha e coluna para o inteiro mais próximo
(54,6; 100,3) ...(55,100)

Exemplo:

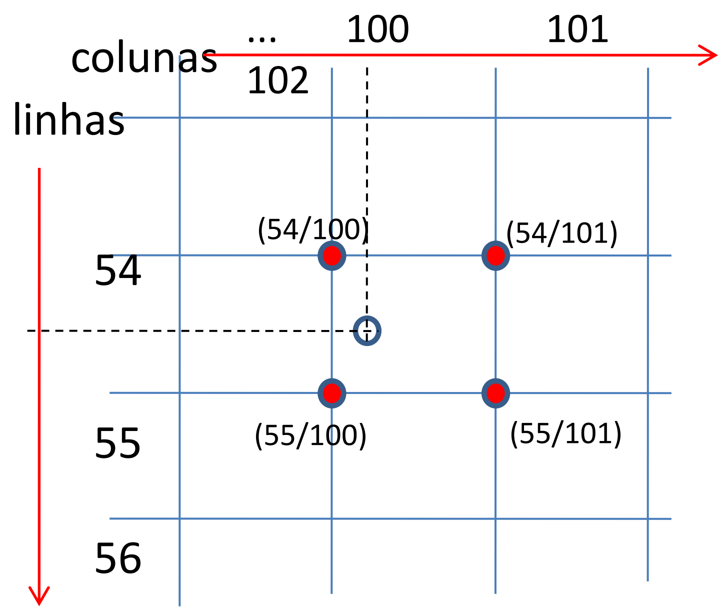


http://northstar-www.dartmouth.edu/doc/idl/html_6.2/Interpolation_Methods.html

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

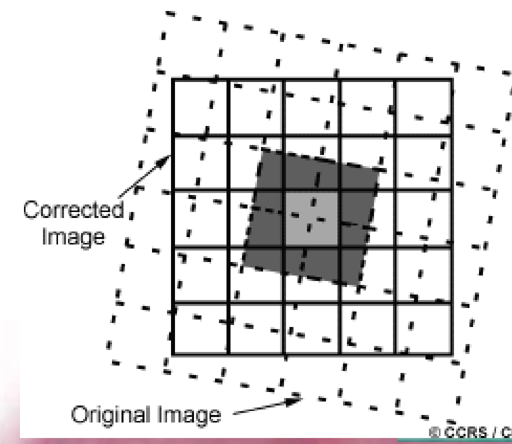


Interpolação bilinear: Consiste em interpolar um novo valor a partir dos quatro vizinhos mais próximos (linha e coluna anterior e posterior).

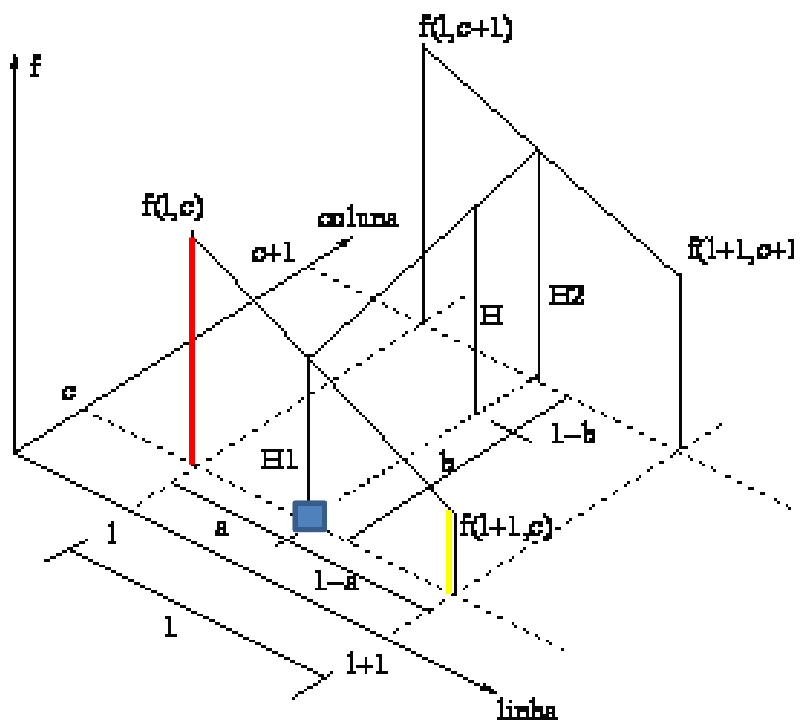


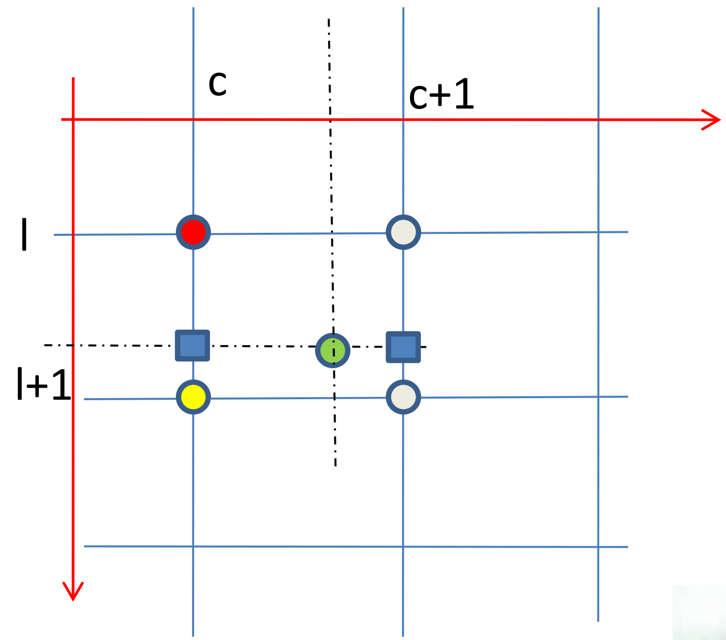
Poderíamos usar o valor médio dos quatro vizinhos, mas isso criaria áreas uniformes quando se muda a escala. Por isso, se interpola com variação dentro desta vizinhança.

Para isto se faz interpolações em linhas e em colunas.

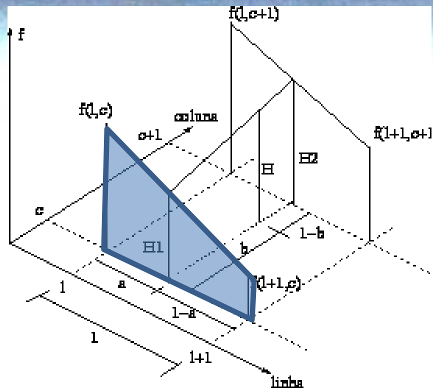


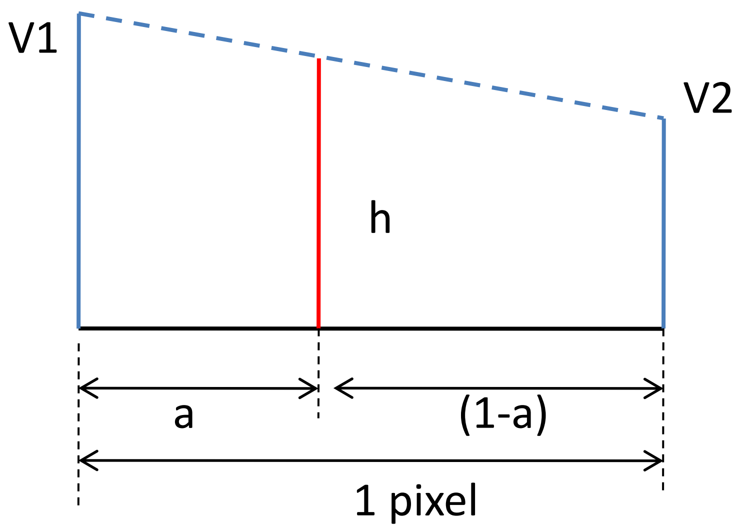
- Interpolação Bilinear

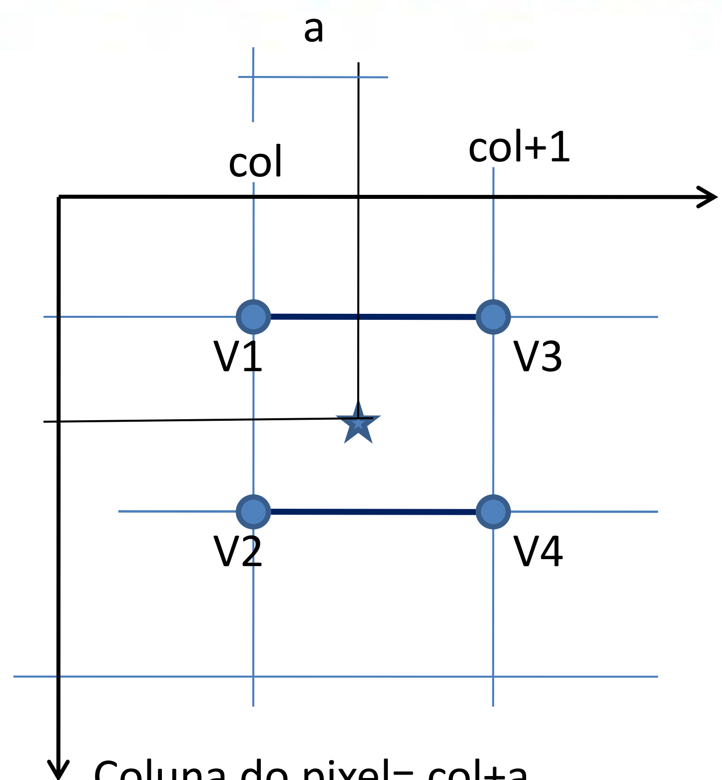




Ao longo de uma linha



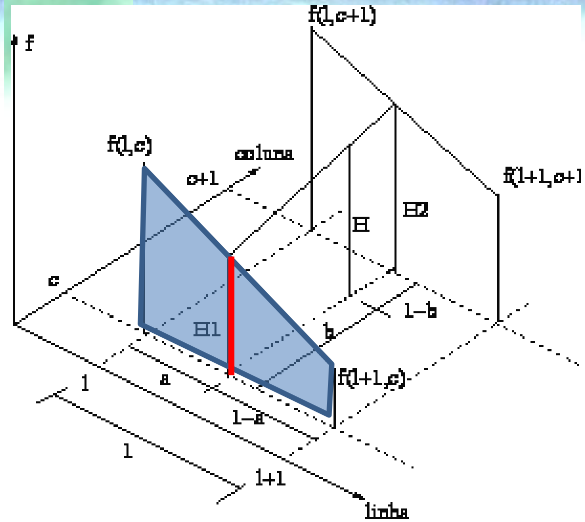


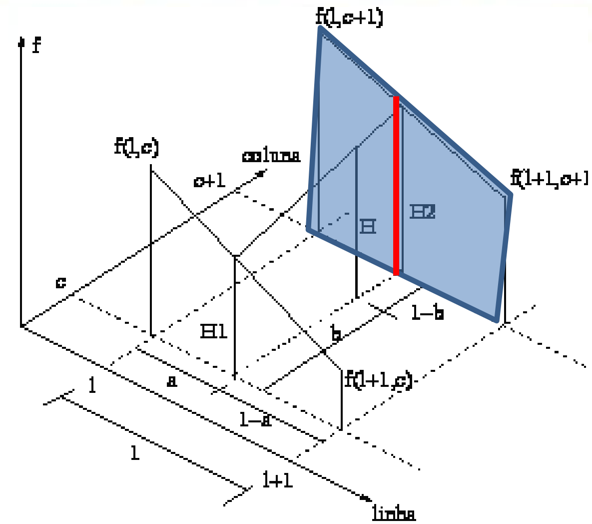


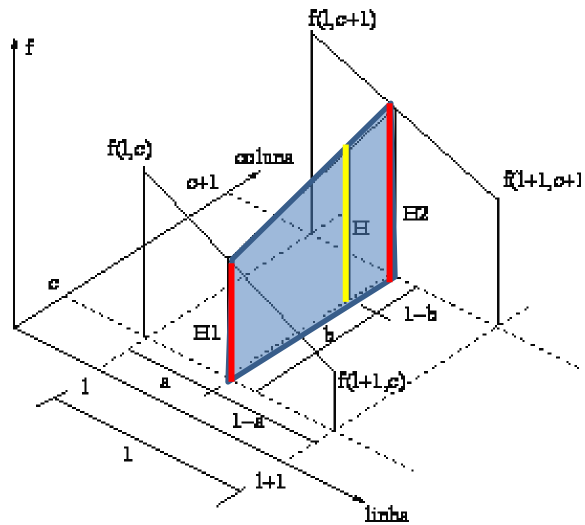
Coluna do pixel= col+a
Ex: 321,81
col=321   a=0,81

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

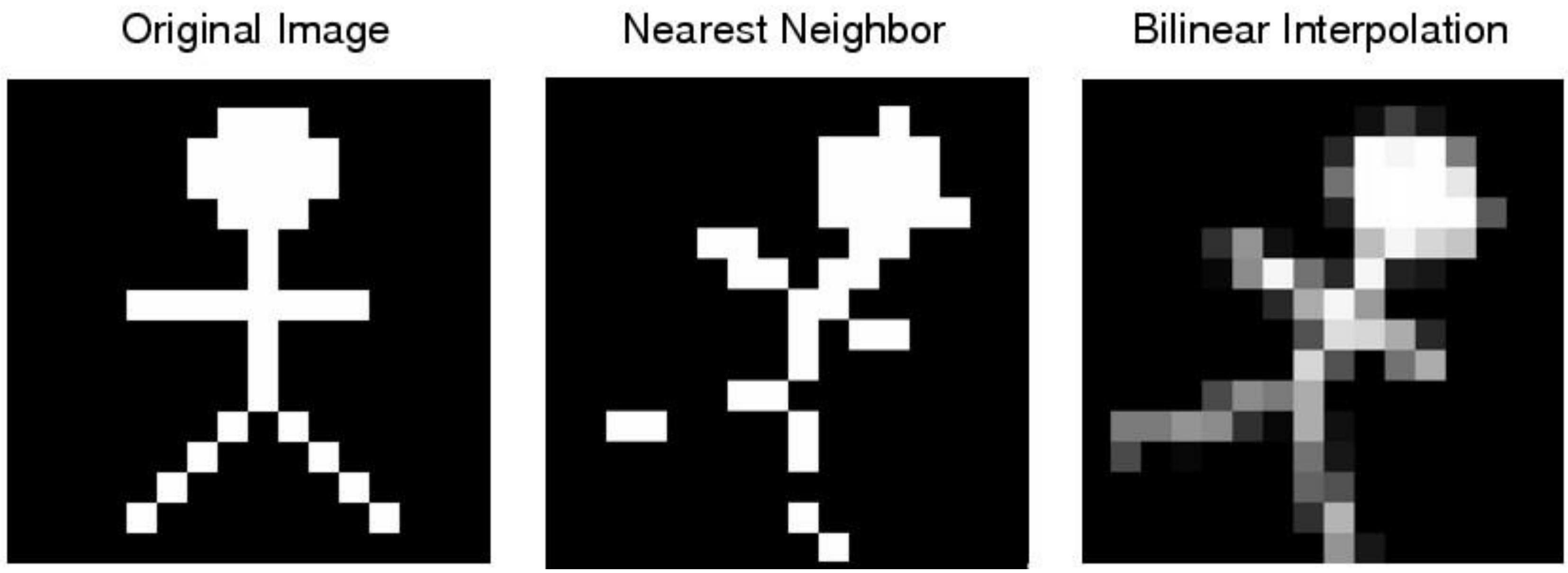
Ao lingo de duas linas e depois ao longo de colunas





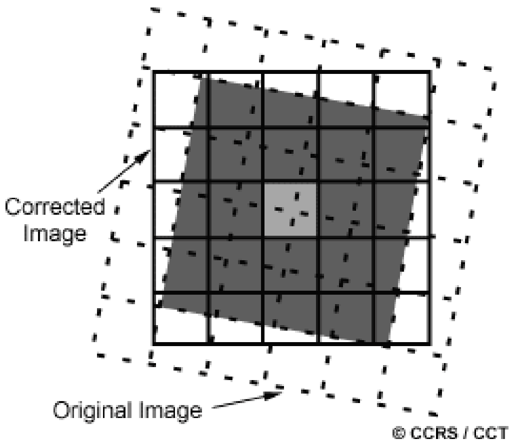


comparação

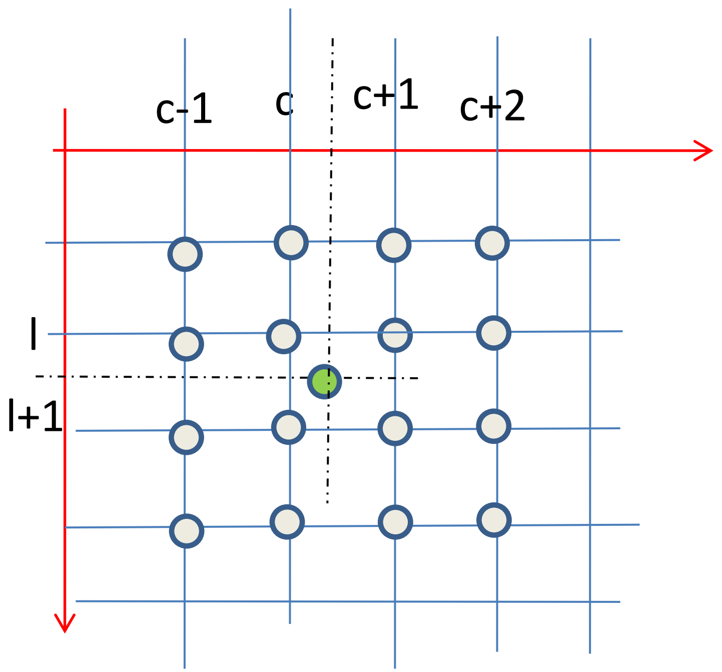


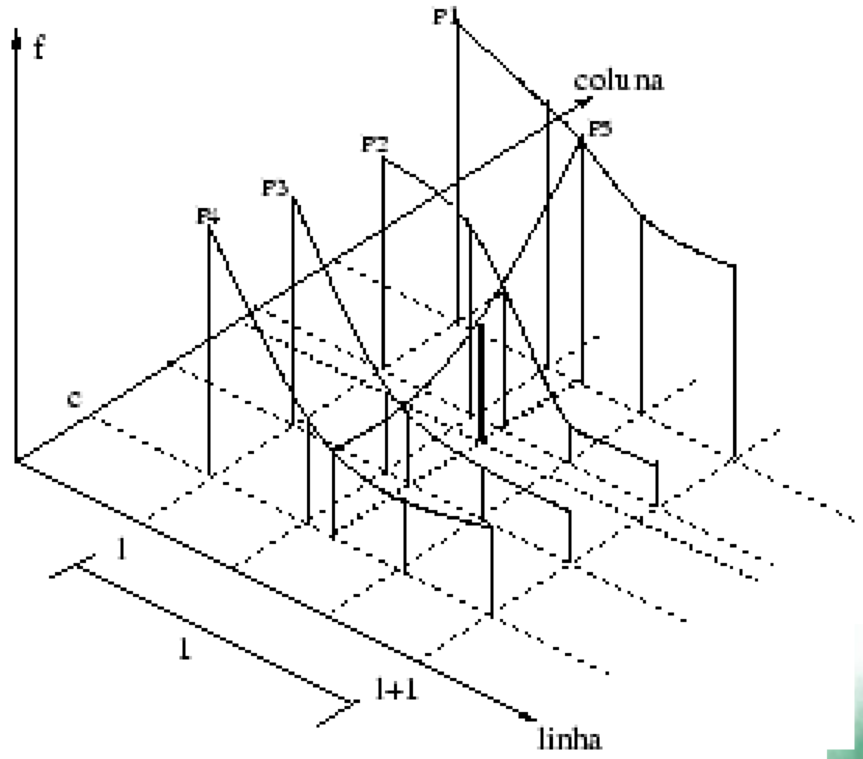
http://northstar-www.dartmouth.edu/doc/idl/html_6.2/Interpolation_Methods.html

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

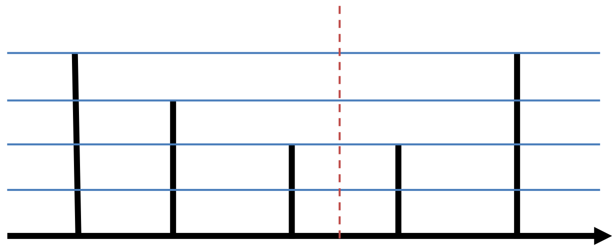


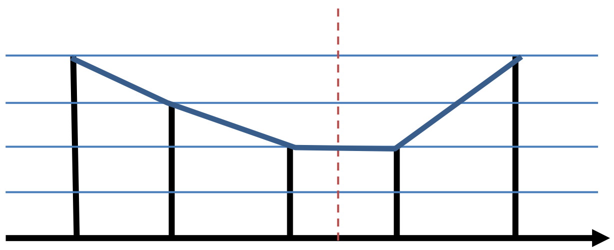
Convolução cúbica. Usa funções cúbicas



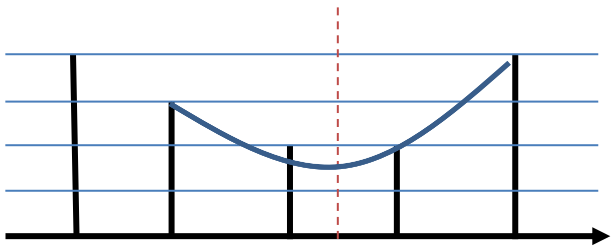


- O que ocorreria se usarmos diferentes interpoladores nesta linha?



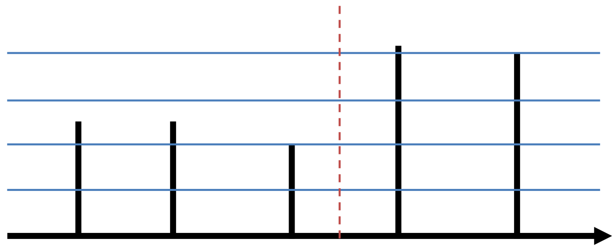


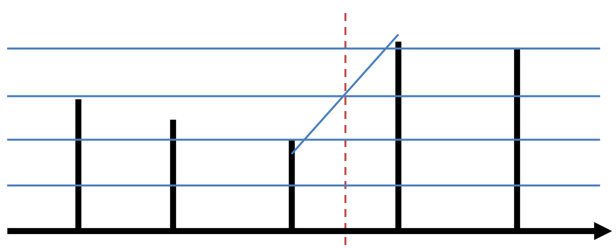
linear



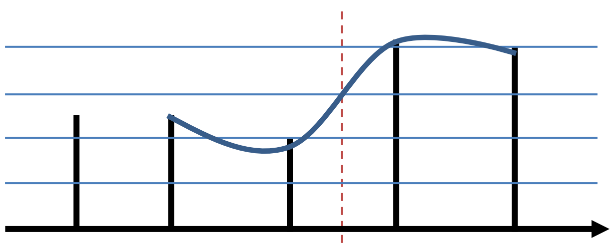
cúbico

- O que ocorreria se usarmos diferentes interpoladores nesta linha?





linear



cúbico

Estimativa de Modelo de Transformação

imagem

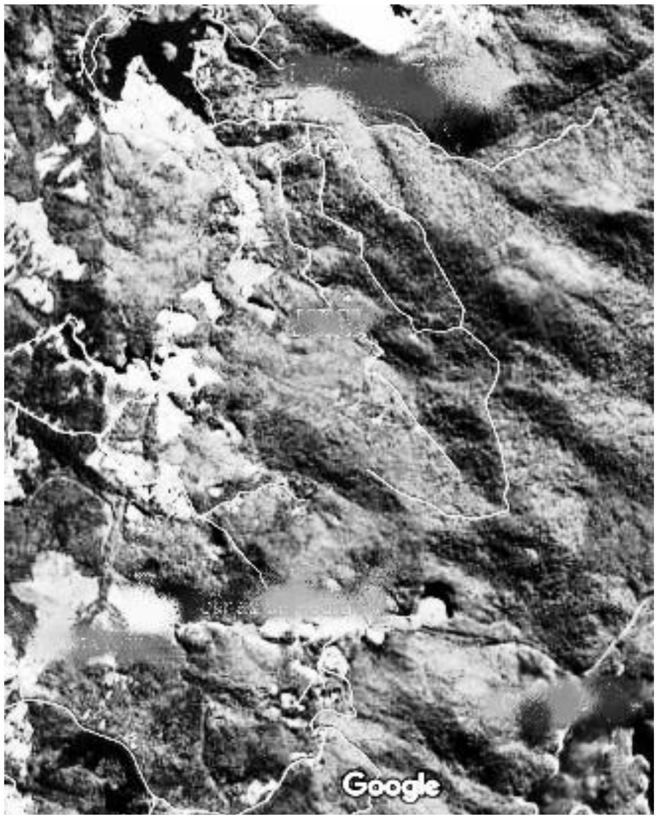
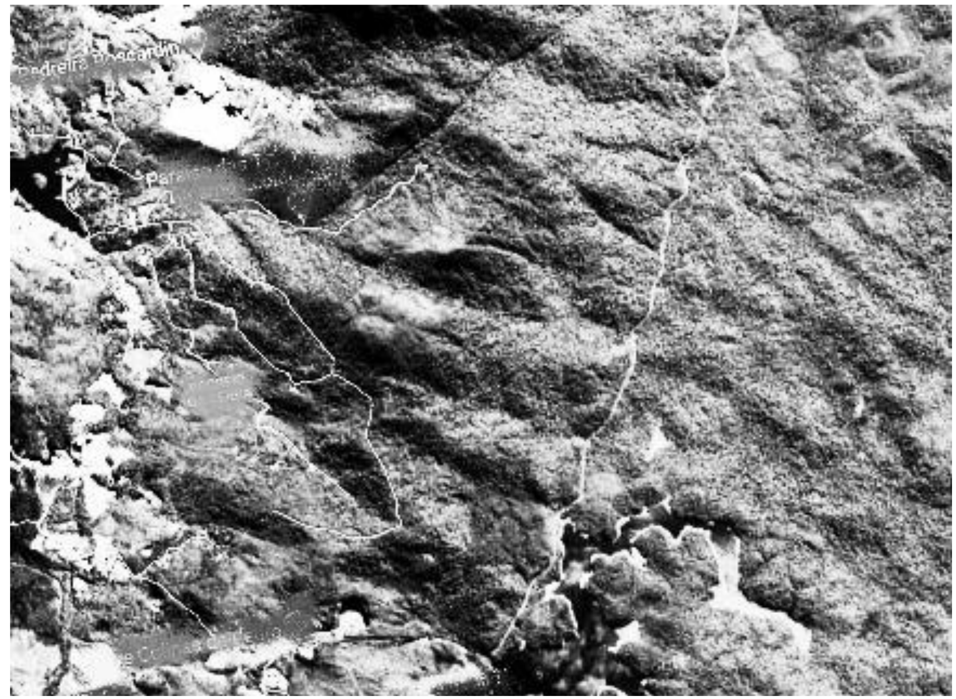


Imagem 2

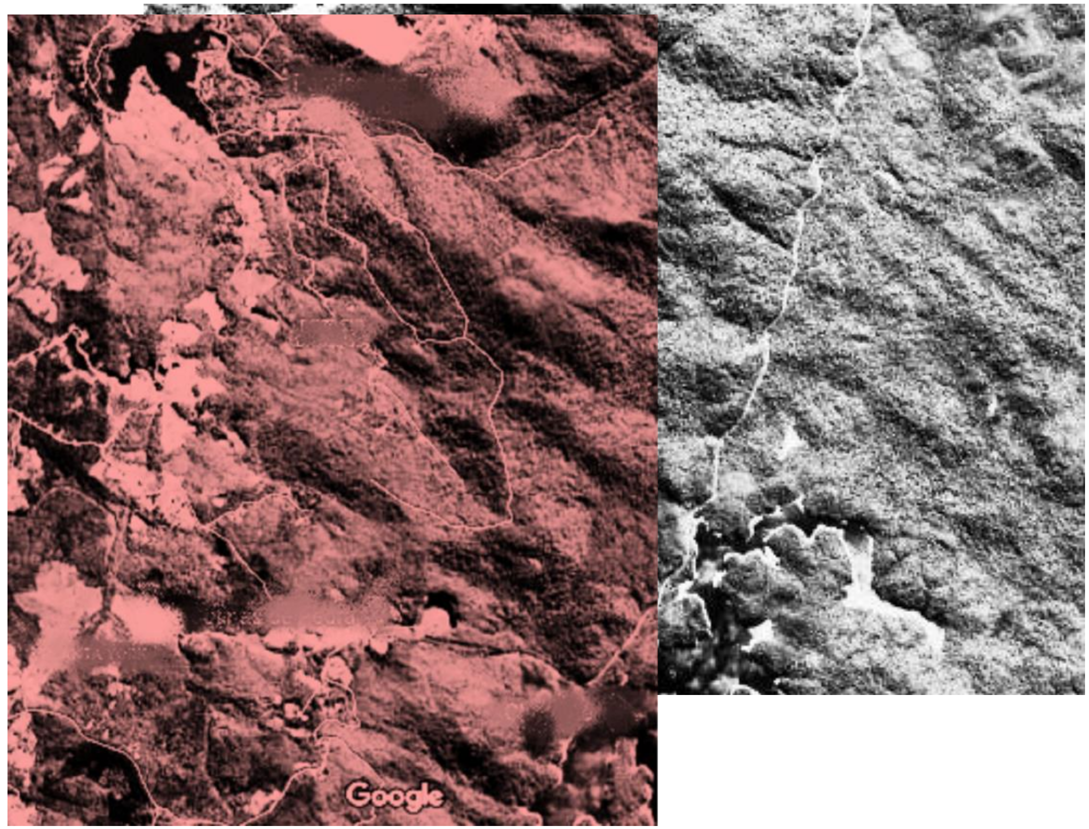


Finalidade:
Transformar a geometria da primeira imagem e ajustá-la à geometria da segunda (base/referencia)

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

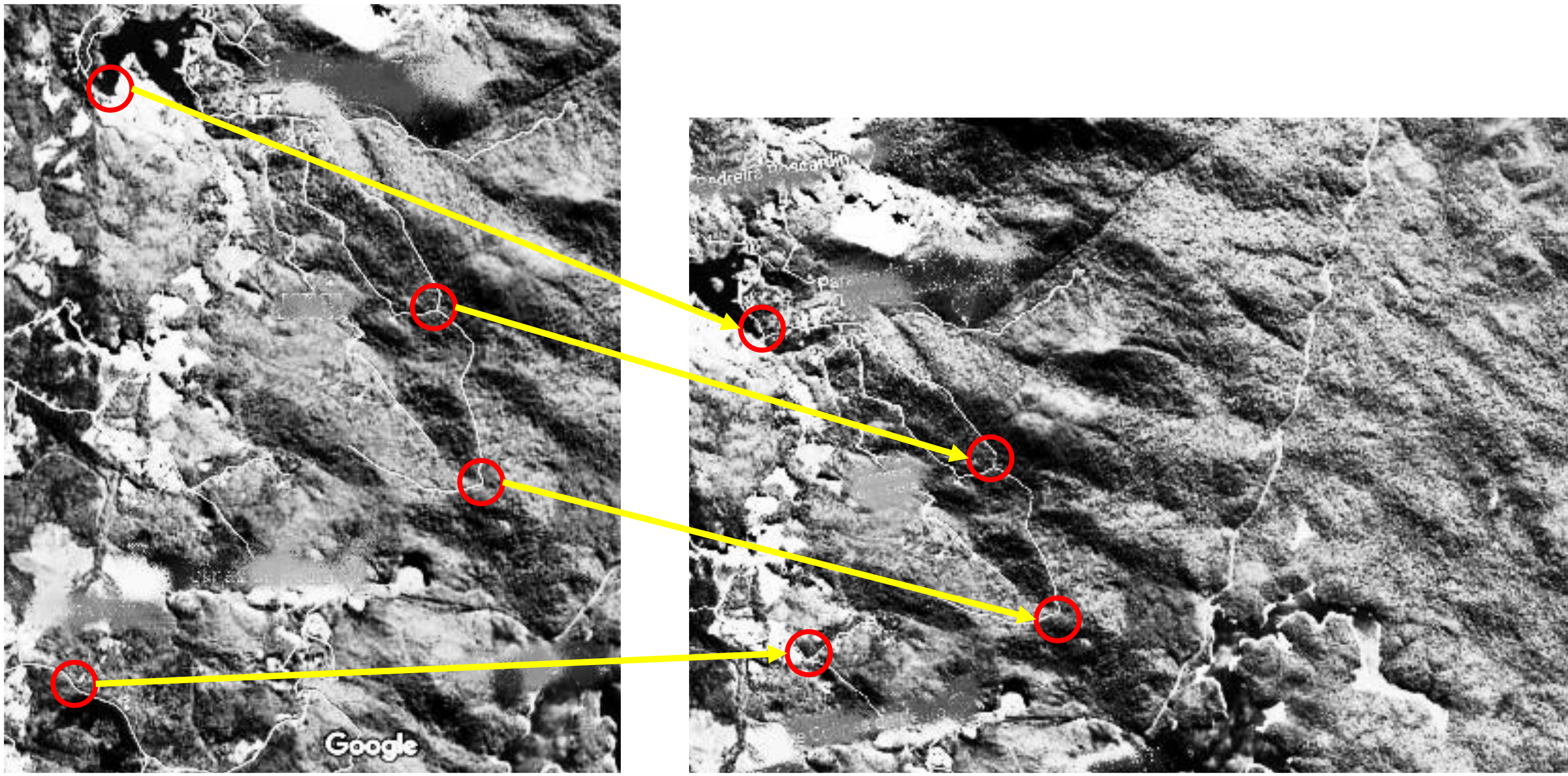


Imagem w

Imagem 2

Coordenadas dos pontos homólogos

	Imagem 1		Imagem 2	
ponto	col	lin	col	lin
1	x1	y1	u1	v1
2	x2	y2	u2	v2
3	x3	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 2xn 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

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

$\Rightarrow$ S= E * A

S(2nx1) = E (2nx6) * A (6x1)

solução

S= E * A [vetor(2nx1) = matriz(2n*6) * vetor(6x1)]

Queremos calcular os parâmetros contidos no vetor “A”

S= E * A

Transposta(E) * S = Transposta(E) * E *A

E’ * S = E’*E*A

E’ * S = (E’*E)*A

Inversa(E’*E)* E’ * S = Inversa(E’*E)* (E’*E)*A

Como inv(M)* M = matriz identidade:

A = 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

```
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
                        # interpolação bilinear (resultado H)
                        H=uint8(H)
                        # copiar o valor no pixel na matriz de saída
                        J(lin, col)=H
```

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
```