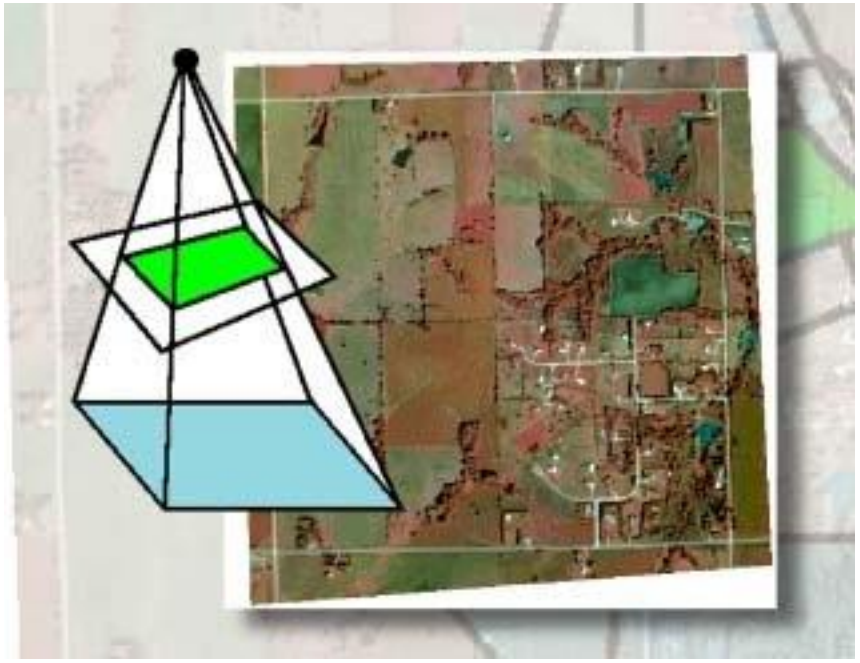
Sensoriamento remoto 1 correções geométricas

Prof. Dr. Jorge Antonio Silva Centeno
Universidade Federal do Paraná

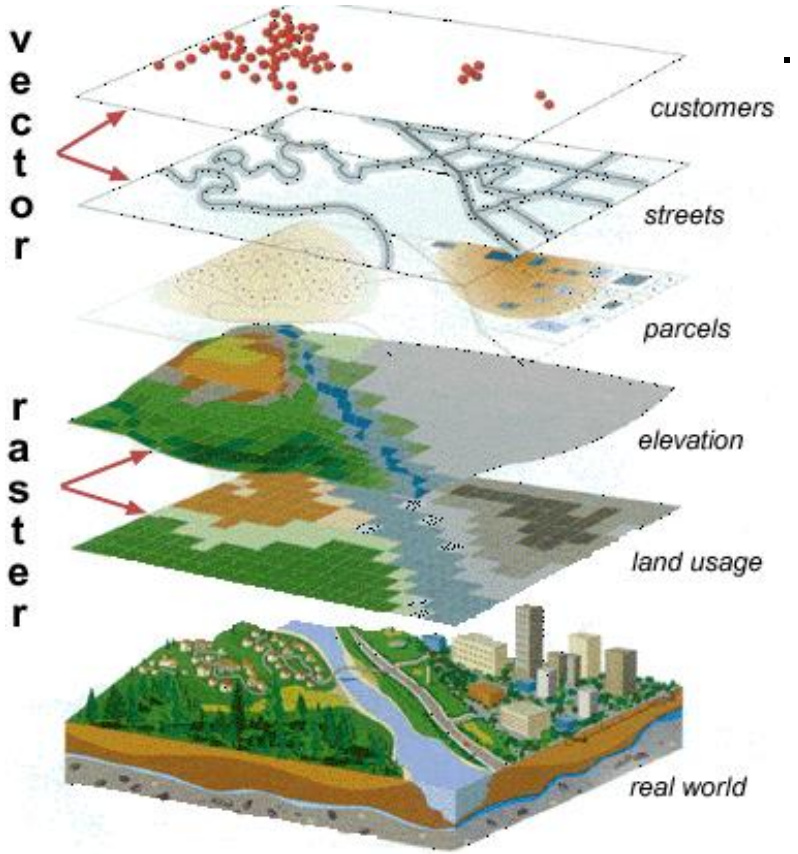
2021

Correções geométricas

Uma imagem digital de sensoriamento remoto é uma representação bidimensional da superfície da Terra, que pode apresentar incoerências geométricas em relação ao mundo real. Isto significa que a posição dos objetos que aparecem na imagem nem sempre pode ser considerada correta.



Correções geométricas



www.seos-project.eu

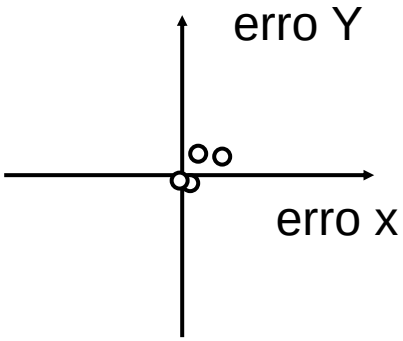
- Delimitar áreas ?
- medir distâncias ?
- SIG?
- => corrigir as distorções

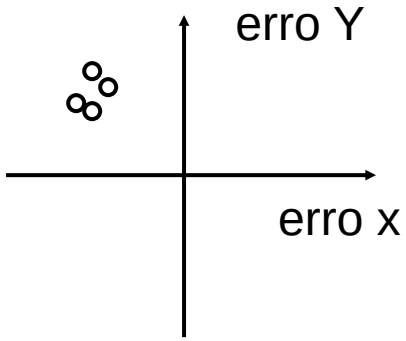
Dois conceitos básicos

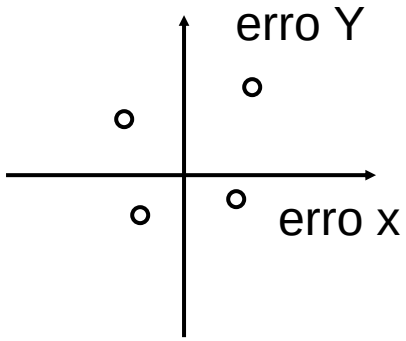
- Acurácia: descreve o grau de similaridade entre as medições e os valores verdadeiros.

- Precisão: Grau de concordância de uma série de observações. Pode ser descrito pelo erro padrão.

Imagem Ikonos II de Guaratuba





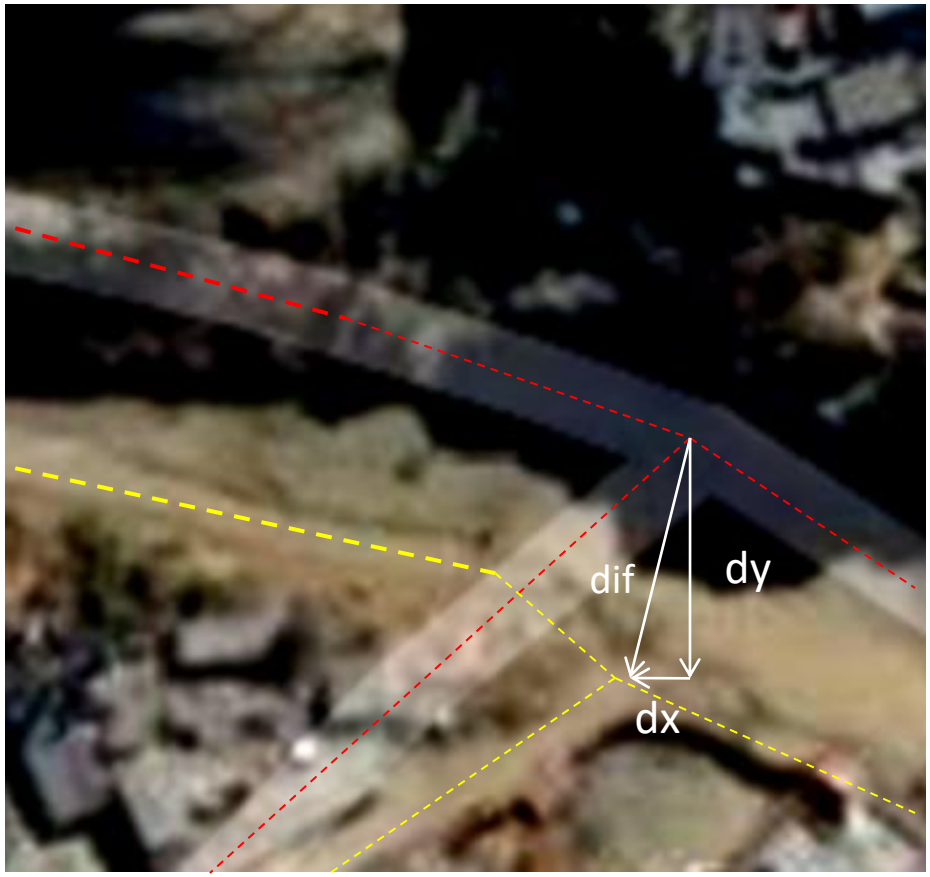


- Acurácia: alta
- Precisão: alta

Exemplo: Erro geométrico Google maps

zoom

diferença



- Com: $dif^2 = dx^2 + dy^2$

Padrão de Exatidão Cartográfica -PEC

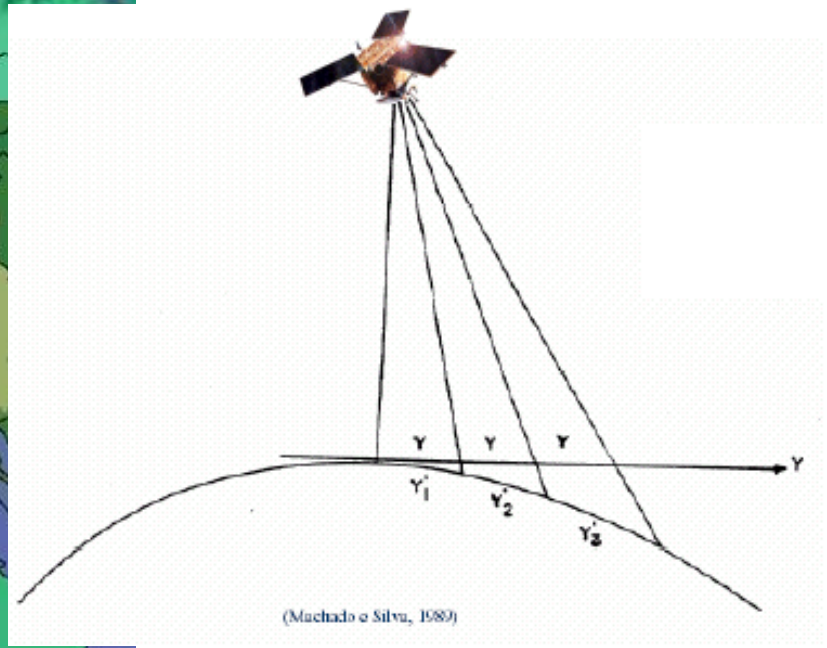
Normas Técnicas da Cartografia Nacional (Decreto nº 89.817, de 20 de junho de 1984)

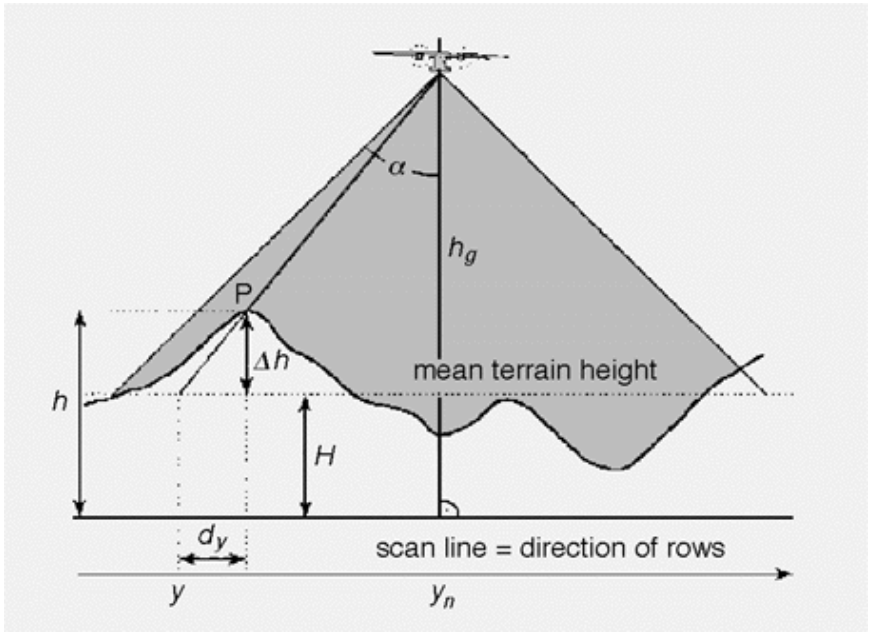
Seção 2 Classes de Cartas
Art.9º
As cartas, segundo sua exatidão, são classificadas nas Classes A, B e C, segundo os critérios seguintes:

Planimétrico	PEC na escala da carta	Erro-Padrão na escala da carta
Classe A	0,5 mm	0,3 mm
Classe B	0,8 mm	0,5 mm
Classe C	1,0 mm	0,6 mm

Fontes de erros geométricos

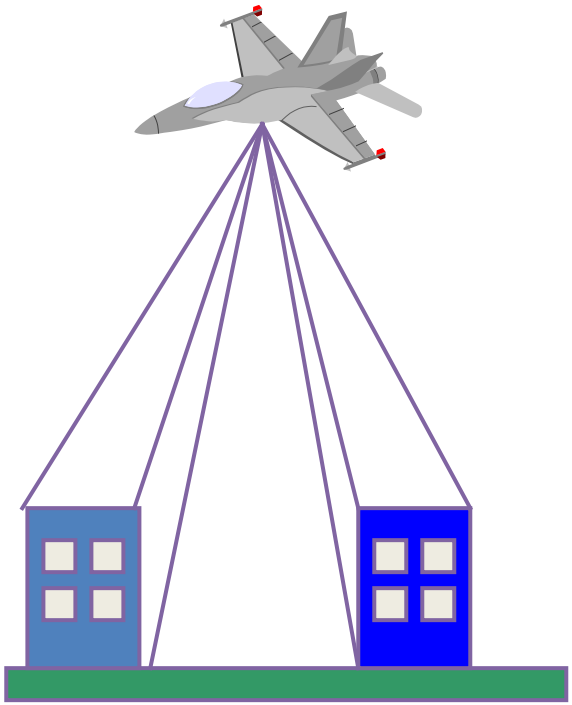
- Curvatura da Terra
- Topografia
- Movimento relativo da superfície da Terra.

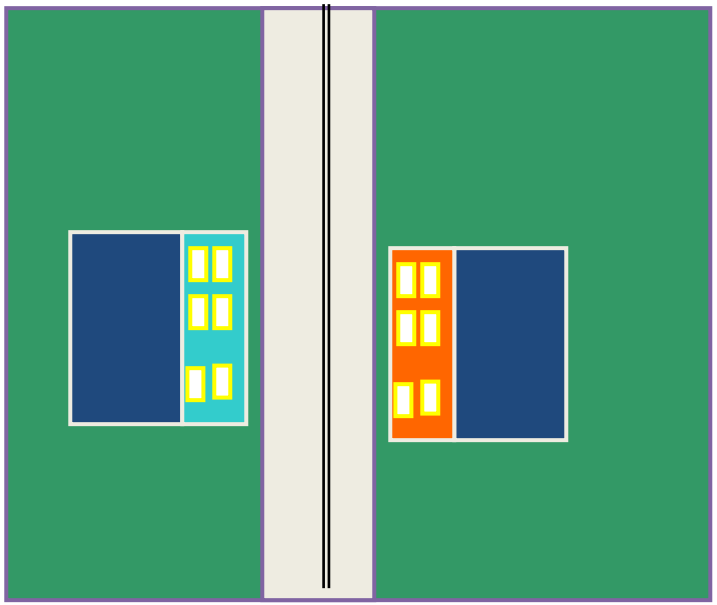




Projeção central.

- Cada pixel é observado sob condições diferentes e a projeção gera deslocamentos relativos diferentes



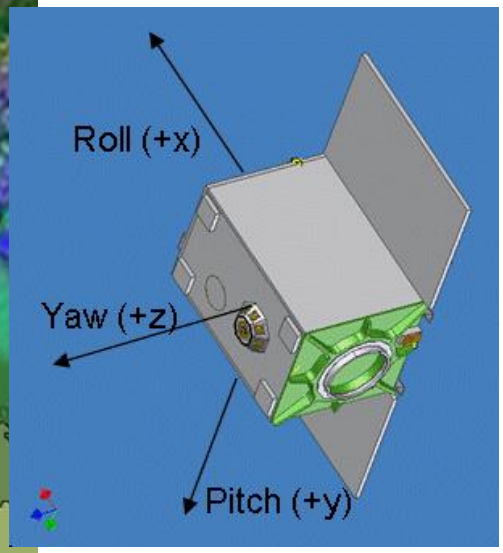


Correções Geométricas

Sistemáticas

Com pontos de controle

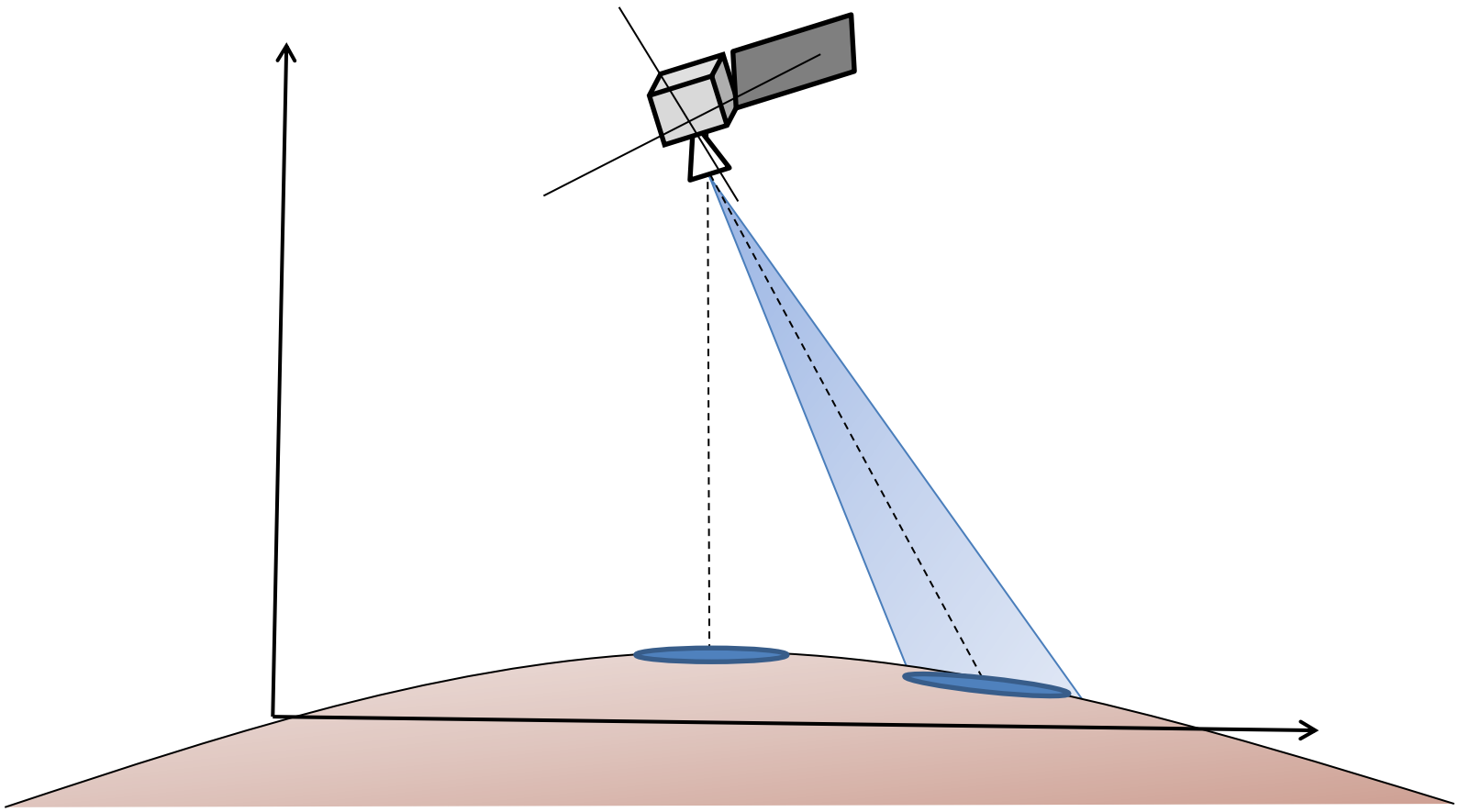
Correção sistemática



Conseguem minimizar as variações espaciais internas presentes na imagem, decorrentes do ângulo de curvatura da terra, variações na velocidade,.

Usando informações presentes no header da imagem: altura e atitude do satélite, deslocamentos de órbita, etc.

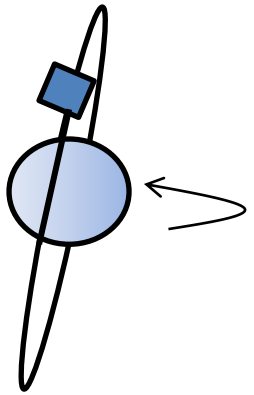
A imagem é associada a um sistema de projeção cartográfica.

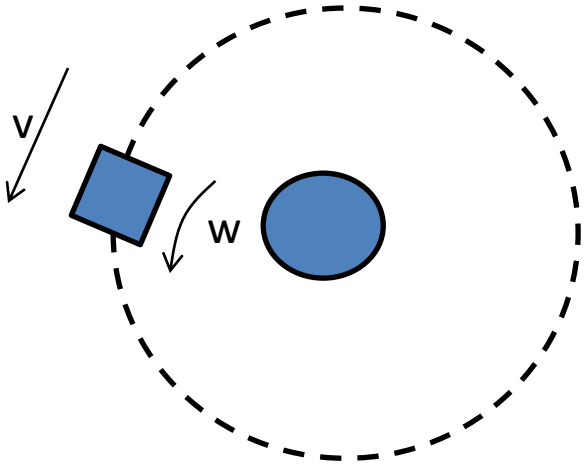


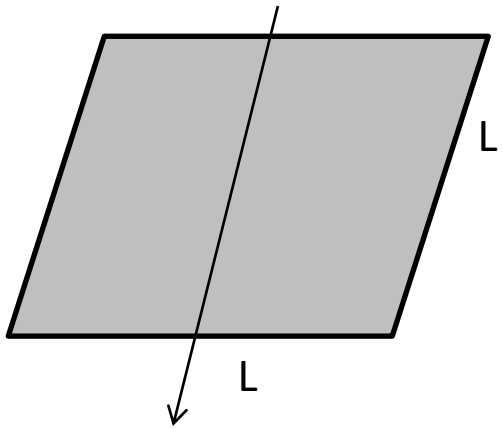
Exemplo de correção sistemática

- Correção do efeito da inclinação relativa da imagem (skew).

Dados:
w_0: velocidade angular do satélite
L: largura/altura da imagem
R: Raio da Terra (6378,16km)

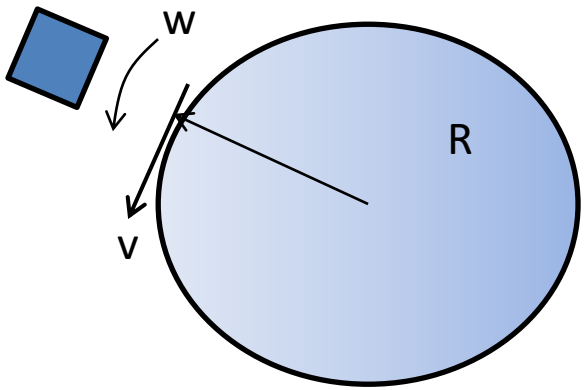


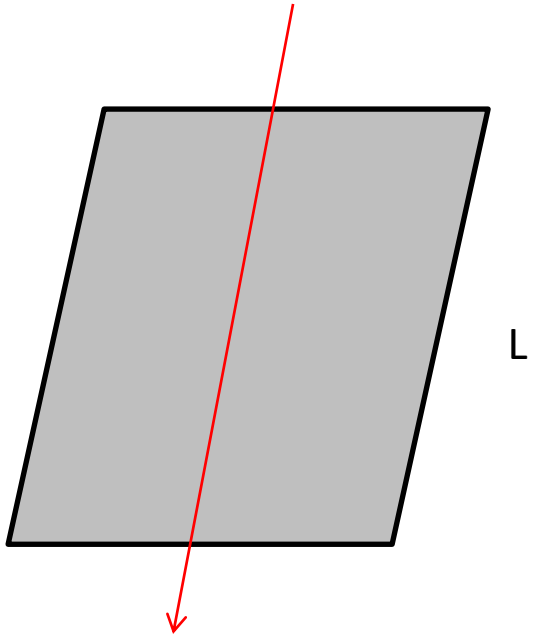




Exemplo de correção sistemática

- Se uma imagem cobre uma área de LxL km, o tempo necessário para o satélite se deslocar do início da imagem até a última linha é:
- $t = L / (w_0 * R)$



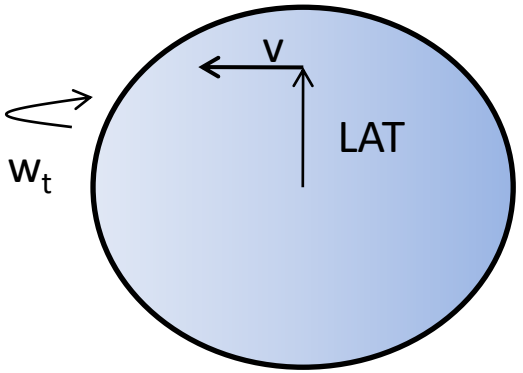


velocidade tangencial na Terra

- A velocidade de rotação da Terra varia em função da latitude. A velocidade tangencial na superfície é:

- $v_t = w_t * R \cos(LAT)$

- LAT=latitude do ponto
- w_t=velocidade angular de rotação da Terra 72,72 10E-6 (rad/seg)
- R: Raio da Terra (6378,16km)

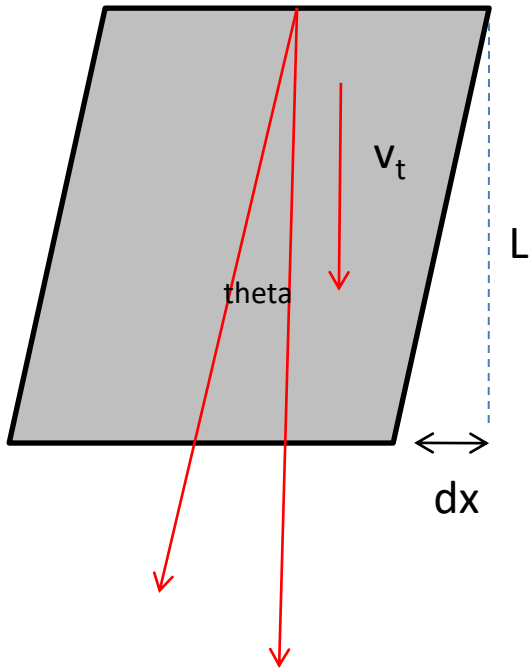


Enquanto o satélite sobrevoa a região coberta pela imagem, a Terra sofre um deslocamento relativo de:

$$dx_0 = v_t * t$$

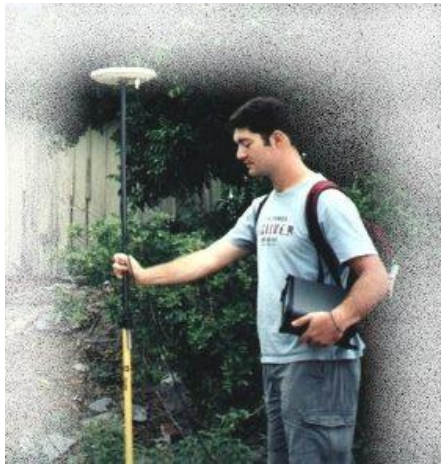
Como o satélite não descreve uma órbita polar, a inclinação da órbita em relação ao eixo norte-sul (theta) deve ser considerada:

$$dx = dx_0 * \cos(theta)$$



Correção com pontos de apoio

- Exige intervenção de um operador.
- A imagem é ajustada com pontos de controle disponíveis em mapa ou com pontos de controle medidos por GNSS.



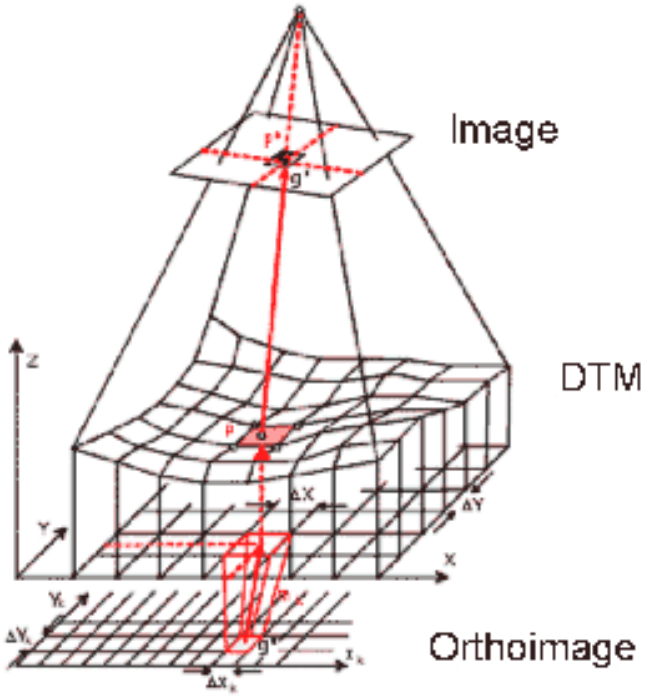
- A precisão geométrica alcançada é de até ½ pixel.
- A qualidade depende dos pontos de apoio e varia em função da região, relevo e disponibilidade de pontos.

Ortoretificação

Exige intervenção de um operador.

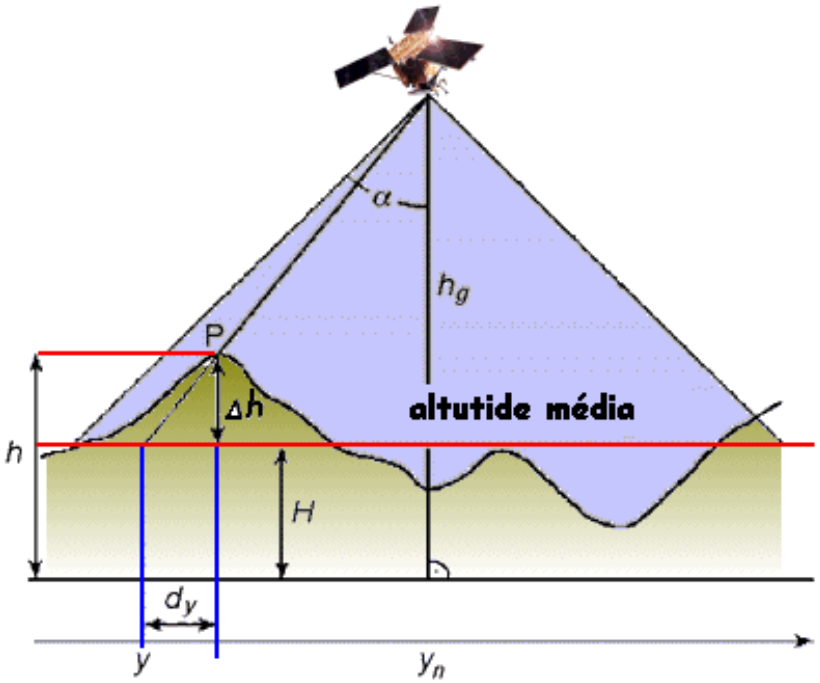
A imagem é corrigida com pontos de controle e usando um Modelo Digital do Terreno (MNT) para corrigir todas as distorções, inclusive aquelas geradas pelo relevo.

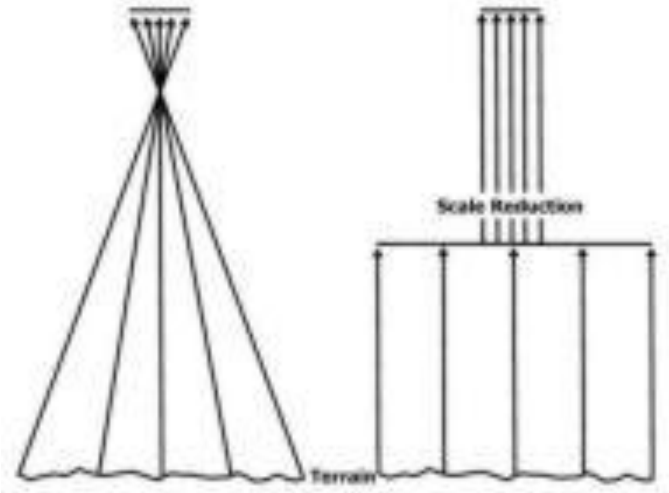
É recomendado que o MNT tenha resolução compatível com a resolução da imagem que está sendo corrigida.



Ortoimagem

- Na imagem obtida com perspectiva central a posição dos objetos depende de sua posição em relação ao sensor, incluindo a topografia.
- A ortocorreção produz uma imagem com projeção ortogonal levando em consideração a variação da topografia. Ara isto, um MDT é necessário.

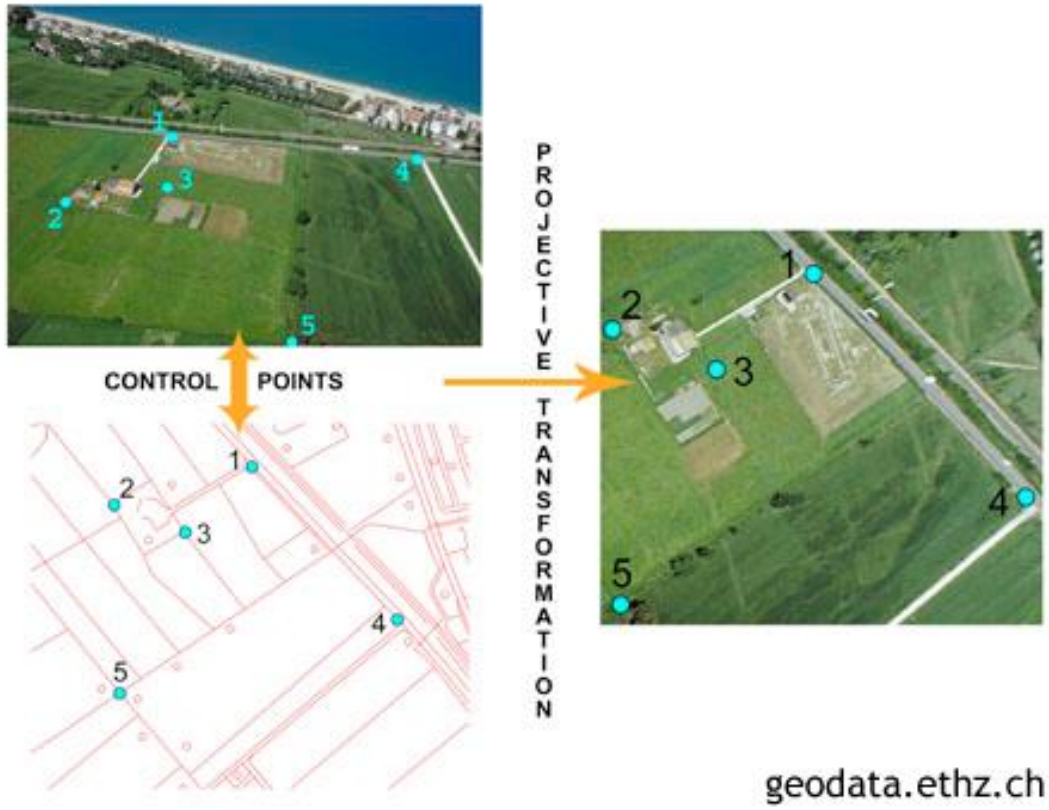




- Correção com pontos de controle

Correção com pontos de controle

- Utiliza pontos de coordenadas conhecidas, visíveis na imagem, para ajustar um modelo das deformações.
- Consiste em ajustar uma transformação geométrica a partir de coordenadas conhecidas e aplicar esta transformação a todos os pontos da imagem
- Resultado: uma imagem livre de erros.



O modelo utilizado

O modelo deve representar a relação entre as linhas e colunas (x,y) da imagem e o sistema local, descrito pelas coordenadas (u,v).

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

ou

$u=f(x,y)$

$v=f(x,y)$

O modelo utilizado na prática é o modelo de polinômio.

A quantidade de parâmetros da transformação depende da complexidade do modelo.

Um modelo simples

Translação

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

Rotação em ângulo q:

Matriz de rotação

$$R = \begin{bmatrix} \cos(q) & -\sin(q) \\ \sin(q) & \cos(q) \end{bmatrix} = \begin{bmatrix} r1 & -r2 \\ r2 & r1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} r1 & -r2 \\ r2 & r1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = R \begin{bmatrix} x \\ y \end{bmatrix}$$

Escala:

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

Escala, rotação e translação - RST

$$\begin{bmatrix} u \\ v \end{bmatrix} = e * \begin{bmatrix} r1 & -r2 \\ r2 & r1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} r1\ e & -r2\ e \\ r2\ e & r1\ e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

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

O modelo utilizado

- O modelo de polinômio de primeira ordem ...6 parâmetros.
- . $x = a_1 * u + a_2 * v + a_3$
- $y = a_4 * u + a_5 * v + a_6$

- modelo de segunda ordem ... 12.

$x = a_1 * u + a_2 * v + a_3 * u * v + a_4 * u^2 + a_5 * v^2 + a_6$

$y = a_7 * u + a_8 * v + a_9 * u * v + a_{10} * u^2 + a_{11} * v^2 + a_{12}$

Direct linear Transform (DLT):

- Considera a variação de altitude dos pontos

$$x = \frac{(a_1*u + a_2*v + a_3*z + a_4)}{(a_9*u + a_{10}*v + a_{11}*z + a_{12})}$$

$$y = \frac{(a_5*u + a_6*v + a_7*z + a_8)}{(a_9*u + a_{10}*v + a_{11}*z + a_{12})}$$

Pontos de apoio

O modelo não é conhecido e deve ser estimado. Para isto, usa-se pontos de apoio, pontos com coordenadas conhecidas na imagem e no terreno ou mapa.

Pontos devem ser criteriosamente escolhidos, para evitar a introdução de novos erros por dificuldade de leitura da posição dos pontos na imagem ou no espaço objeto.

Cuidados:

Os pontos devem ser facilmente identificáveis na imagem e no terreno.

A distribuição espacial dos pontos deve cobrir a área inteira de maneira uniforme.

A quantidade ideal de pontos depende do modelo.

- Certo

Errado:
Pontos concentrados e alinhados

- Para a estimativa dos parâmetros, o método dos mínimos quadrados é usado
- . (x,y) = f(u,v) ou
- . (u,v) = f(x,y)
- A estimativa é baseada nas coordenadas dos pontos de apoio:

Pto.	coluna	linha	L_{mapa}	N_{mapa}
1a	15	35	7.099.071	67.908.765
2a	2056	1870	7.609.000	67.080.005
3a	809	124	7.293.087	67.800.001
4b	Etc, etc			

Estimativa do modelo

a) propor modelo. Ex. modelo linear de primeira ordem:

- $x = a_1 * u + a_2 * v + a_3$
- $y = a_4 * u + a_5 * v + a_6$

b) Coletar suficientes pontos de apoio com coordenadas x,y e u,v conhecidas.

c) Estimar os parâmetros do modelo (ajustamento)

d) Verificar do modelo com pontos de verificação

Para cada ponto, um conjunto de duas equações é obtido:

para N pontos:

$$x1 = a1 * u1 + a2 * v1 + a3 + 0 + 0 + 0$$

$$y1 = 0 + 0 + 0 + a4 * u1 + a5 * v1 + a6$$

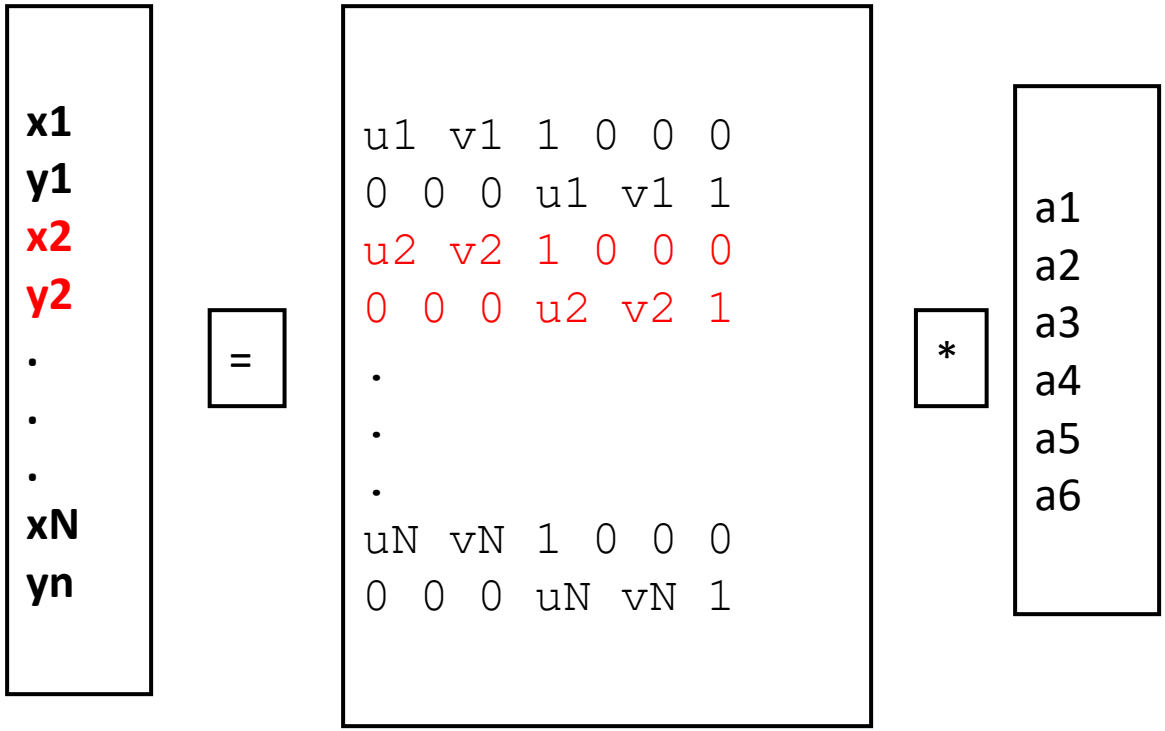
$$x2 = a1 * u2 + a2 * v2 + a3 + 0 + 0 + 0$$

$$y2 = 0 + 0 + 0 + a4 * u2 + a5 * v2 + a6$$

...

$$xN = a1 * uN + a2 * vN + a3 + 0 + 0 + 0$$

$$yN = 0 + 0 + 0 + a4 * uN + a5 * vN + a6$$



Ou na forma matricial: $T = W * A$

Solução

T = W * A, *procura-se A*

- Opção 1:

  A=T/W

Opção 2:

T = W * A

inv(W)*T=[inv(W)*W]* A

A= inv(W)*T

Opção 3:

T = W * A

W^t *T = [W^t * W]*A

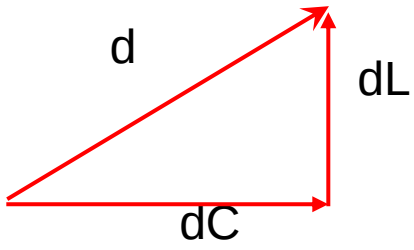
inv([W^t * W]) W^t *T = inv([W^t * W]) [W^t * W]*A

A= inv([W^t * W]) W^t *T

Qualidade do modelo

- Os parâmetros são estimados e a qualidade do modelo é avaliada, analisando os resíduos:

Pto.	col	lin	L_{mapa}	N_{mapa}	Col est	Lin est	dCol	dLin	d
1a	15	35	7.099.071	67.908.765	14,5	35	-0,5	0	0,5
2a	2056	1870	7.609.000	67.080.005	2056	1869	0	-1	1
3a	809	124	7.293.087	67.800.001	805	125	-4	1	4,12
4b	Etc, etc								

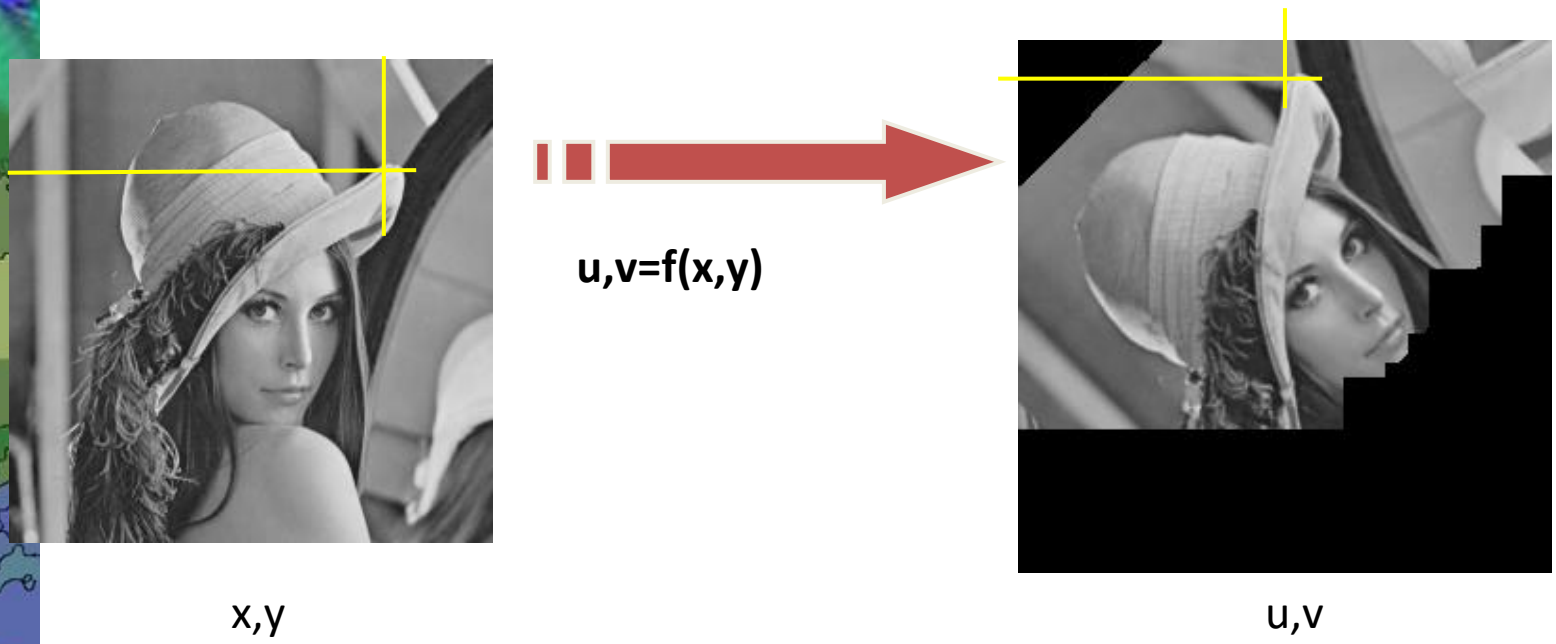


Erro médio

Como preencher a imagem nova?

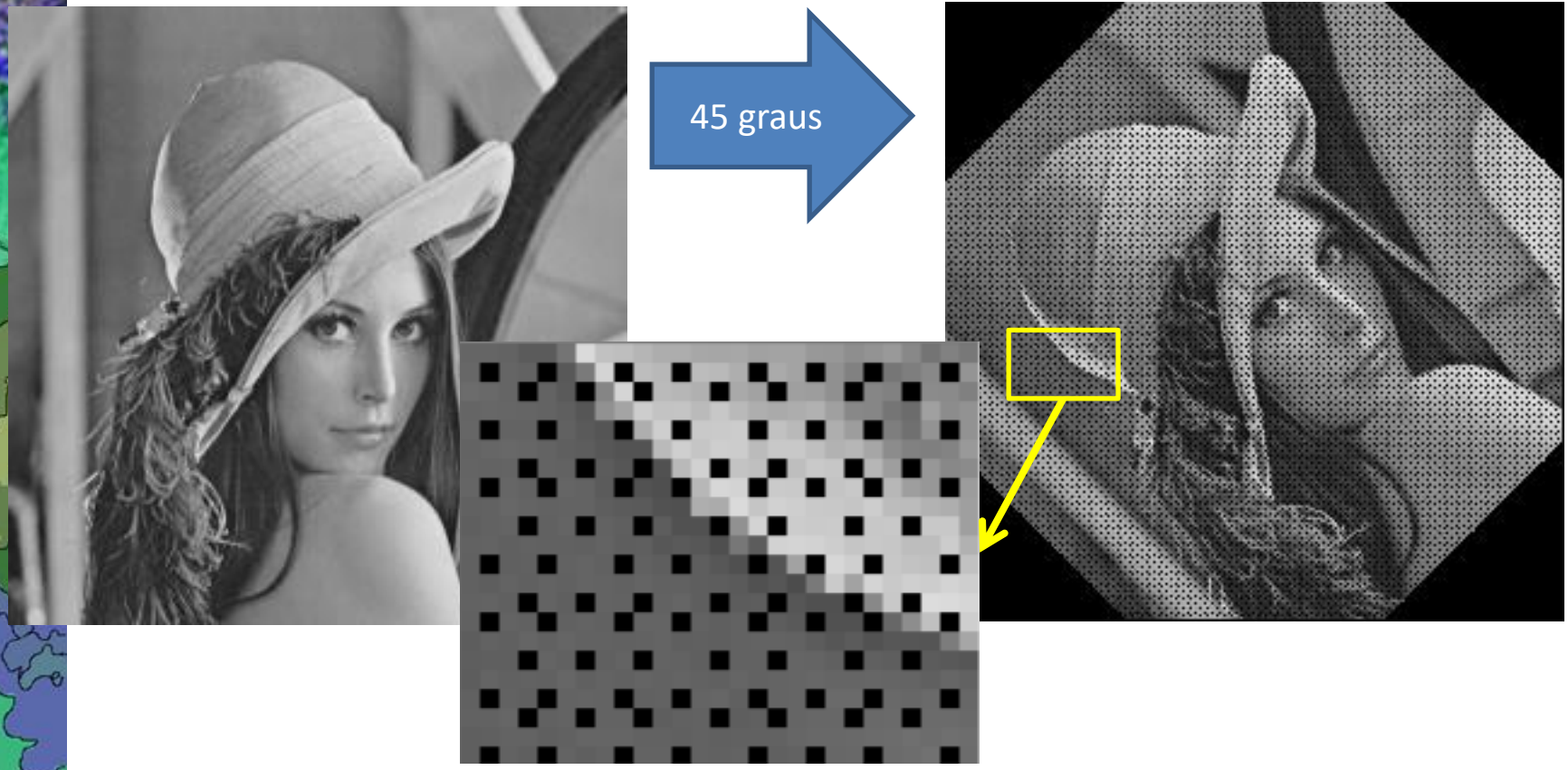
Mapeamento direto

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



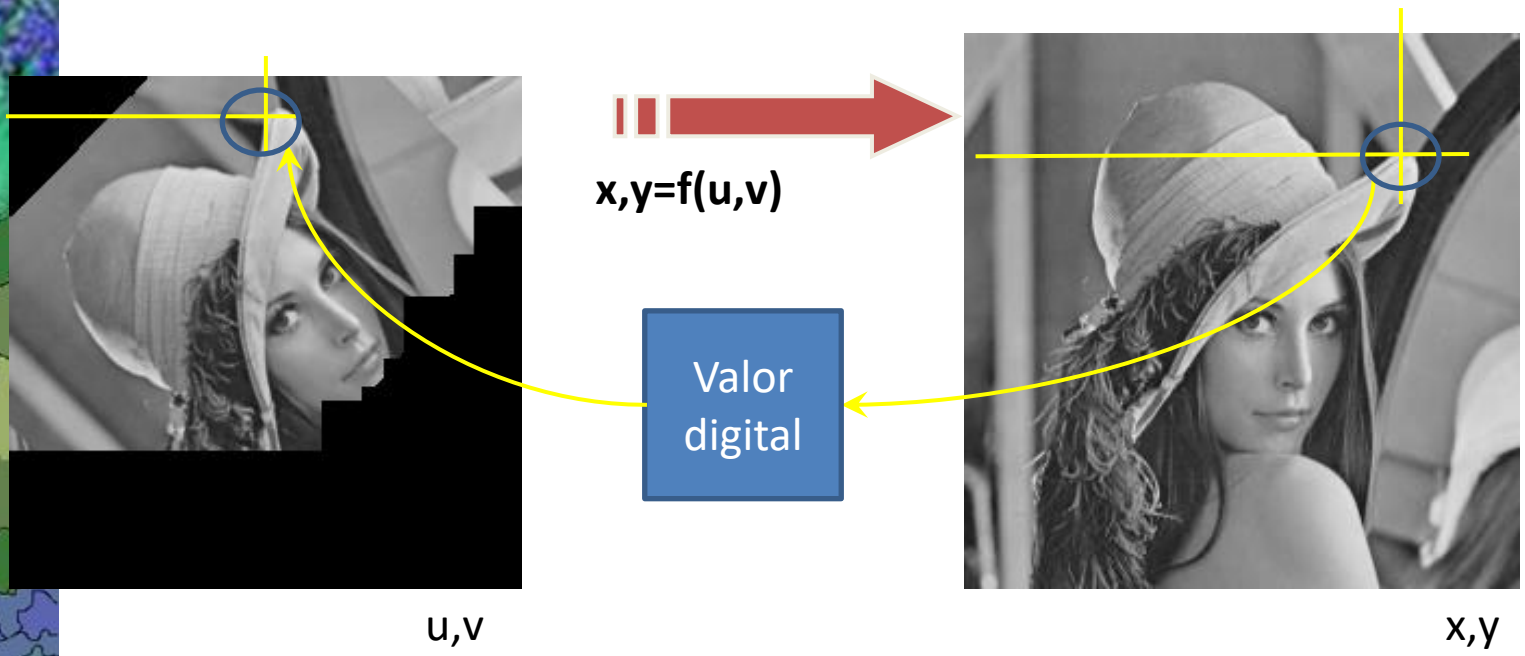
falhas

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



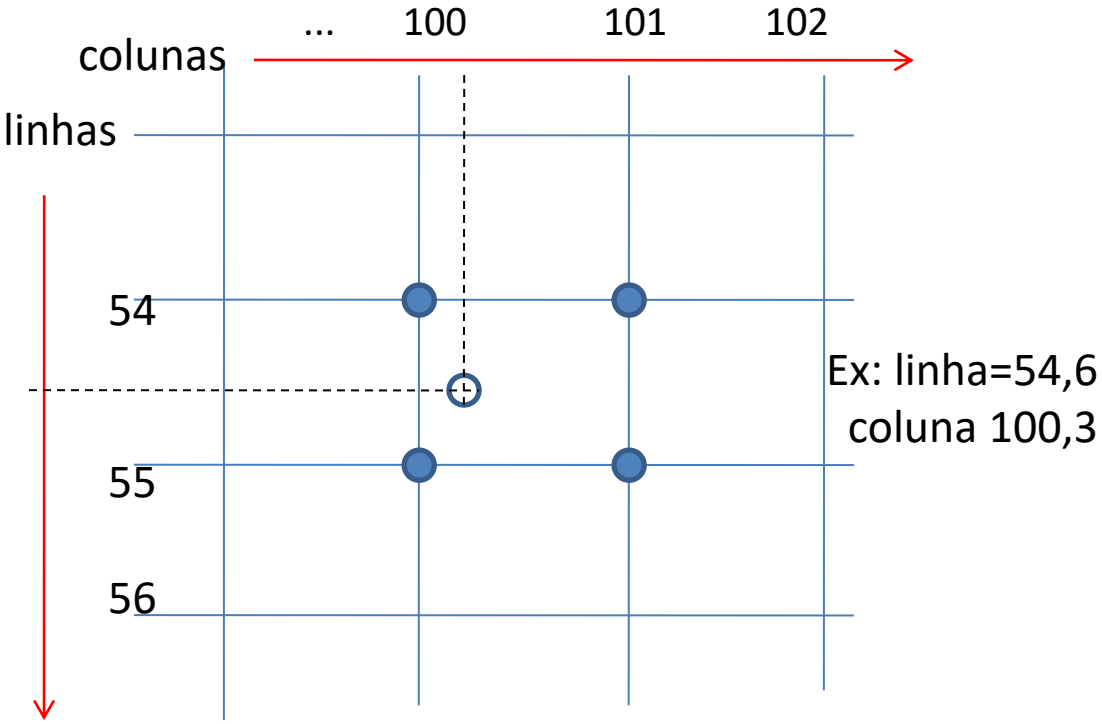
Mapeamento inverso

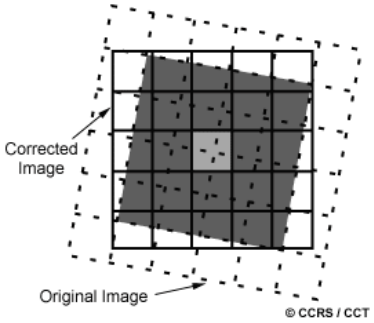
- Dadas as coordenadas da imagem de saída, calcular a posição na imagem de entrada e copiar o valor digital.



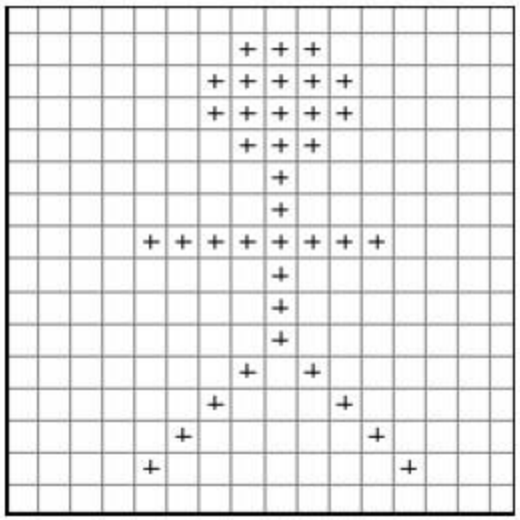
problema

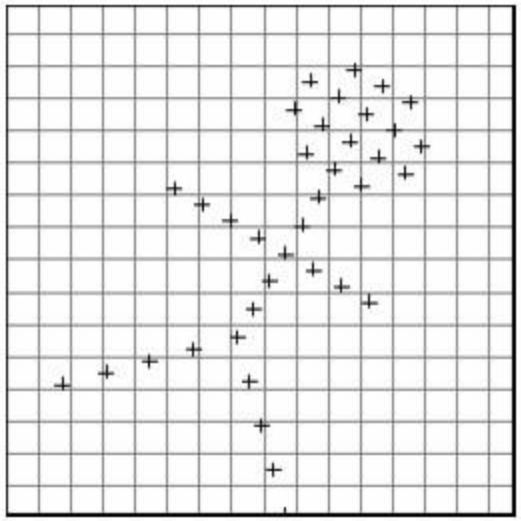
- As coordenadas calculadas nem sempre correspondem a números inteiros e por este motivo o novo valor digital deve ser interpolado.





Exemplo:

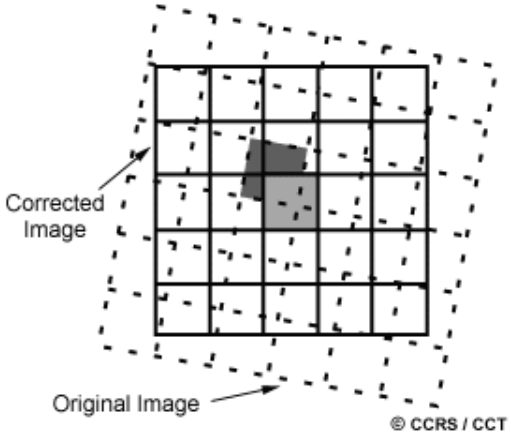




http://northstar-www.dartmouth.edu/doc/idl/html_6.2/Interpolation_Methods.html

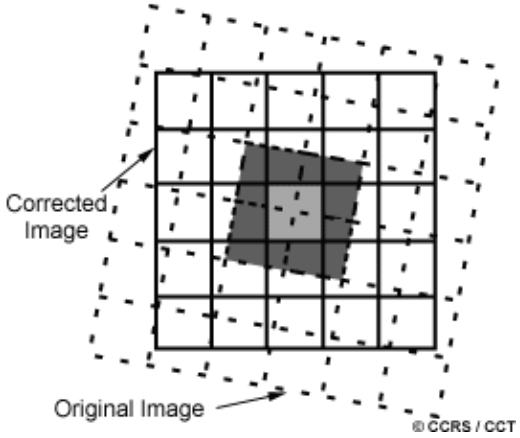
vizinho mais próximo:

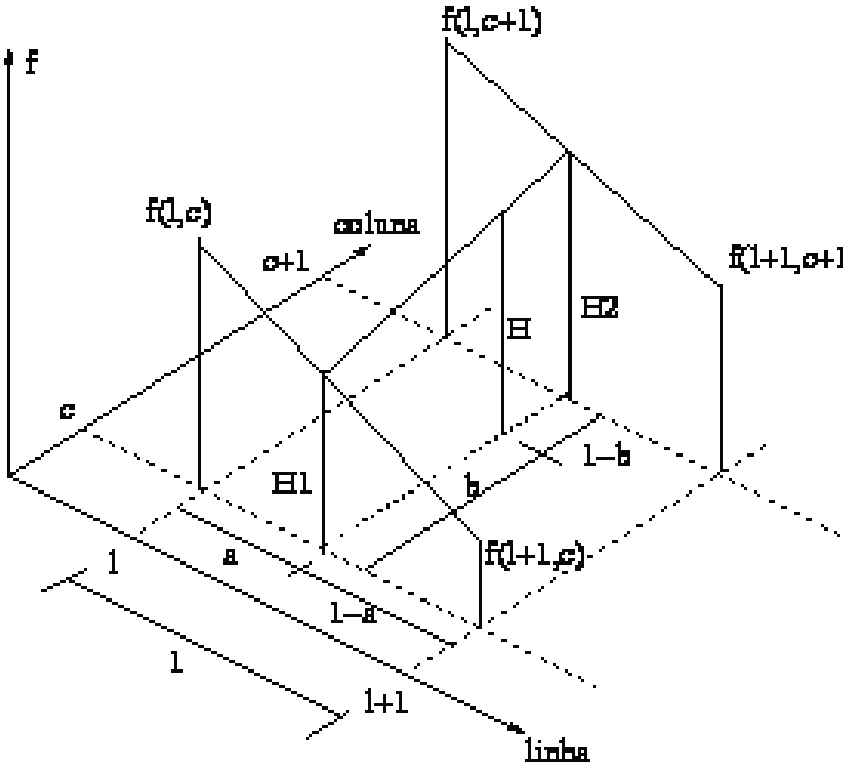
Consiste na escolha do valor do contador digital do pixel mais próximo. Não gera novos valores interpolados. Produz um efeito de degrau em imagens de nível de cinza, devido ao arredondamento da posição do pixel na imagem original. (imagens temáticas?)



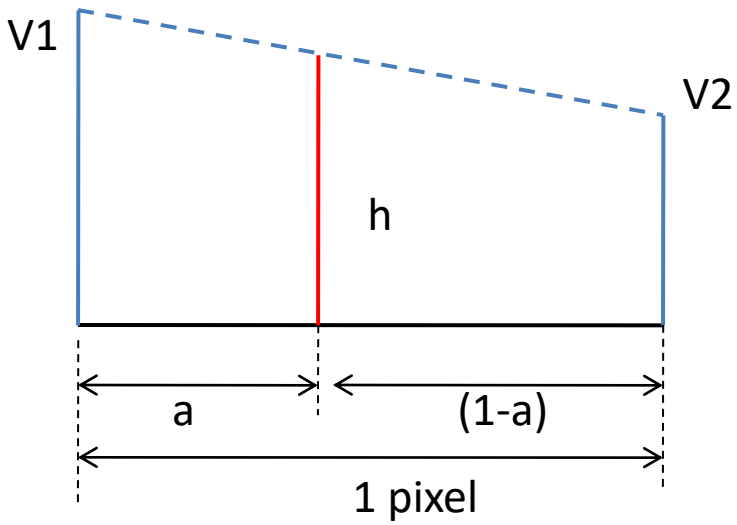
Interpolação bilinear:

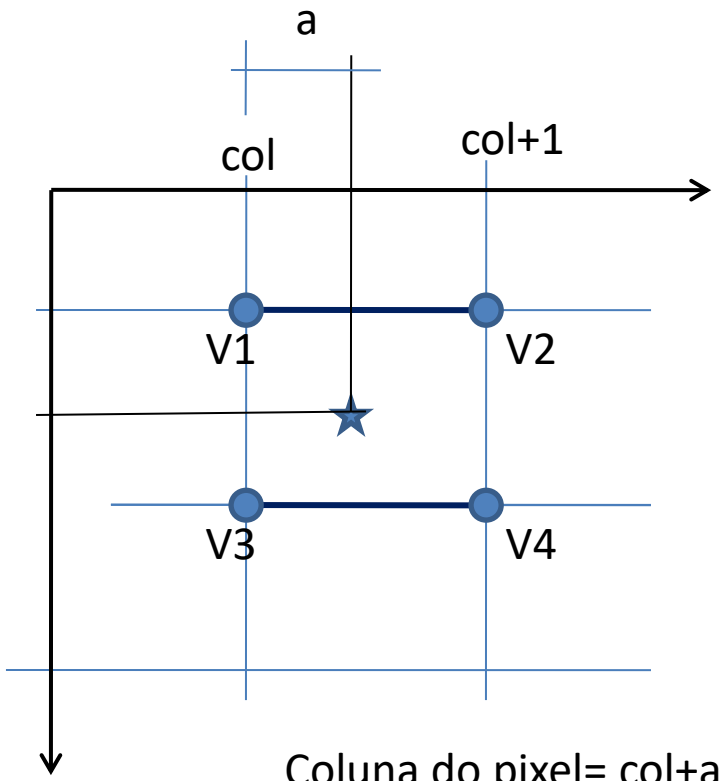
- Consiste em interpolar um novo valor a partir dos quatro vizinhos mais próximos (linha e coluna anterior e posterior).





Ao longo de uma linha





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

Se V1=100 eE V2=30, h=?

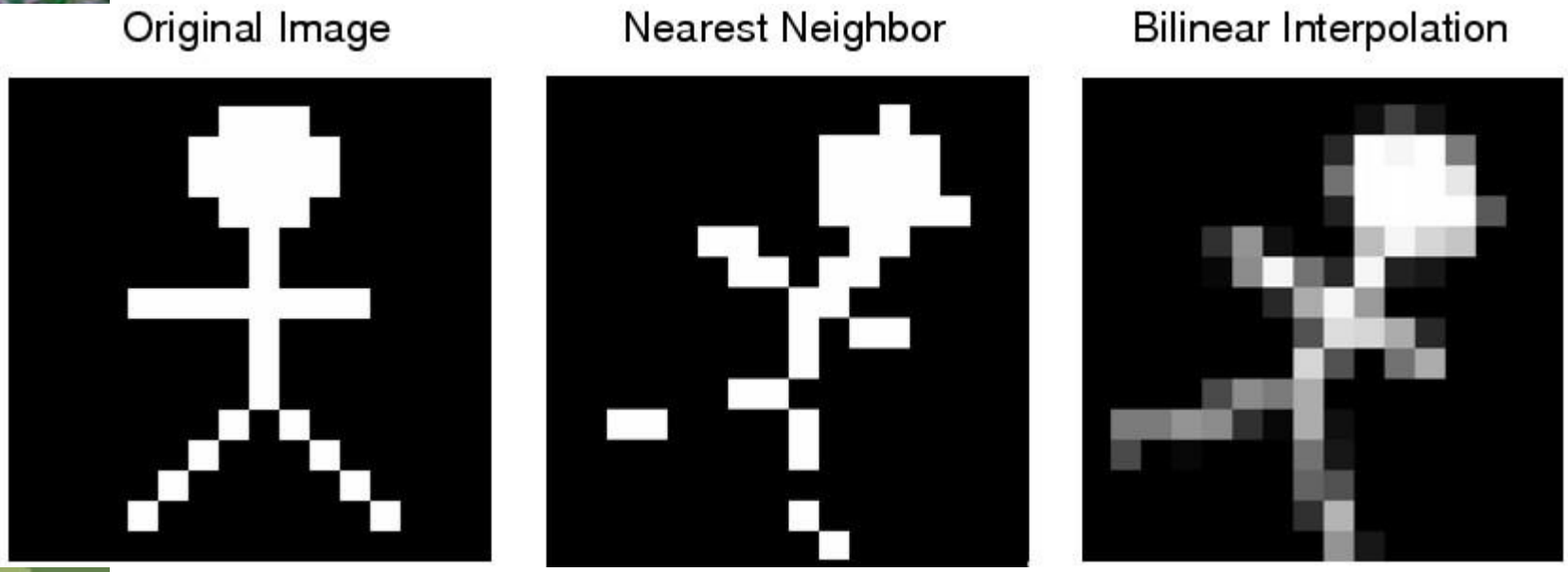
Coluna do pixel= col+a

Ex: 321,81

col=321

a=0,81

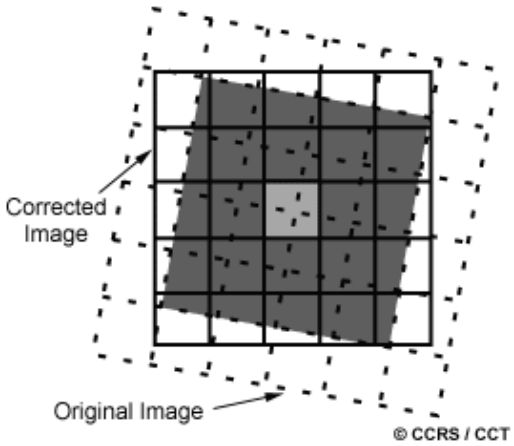
comparação

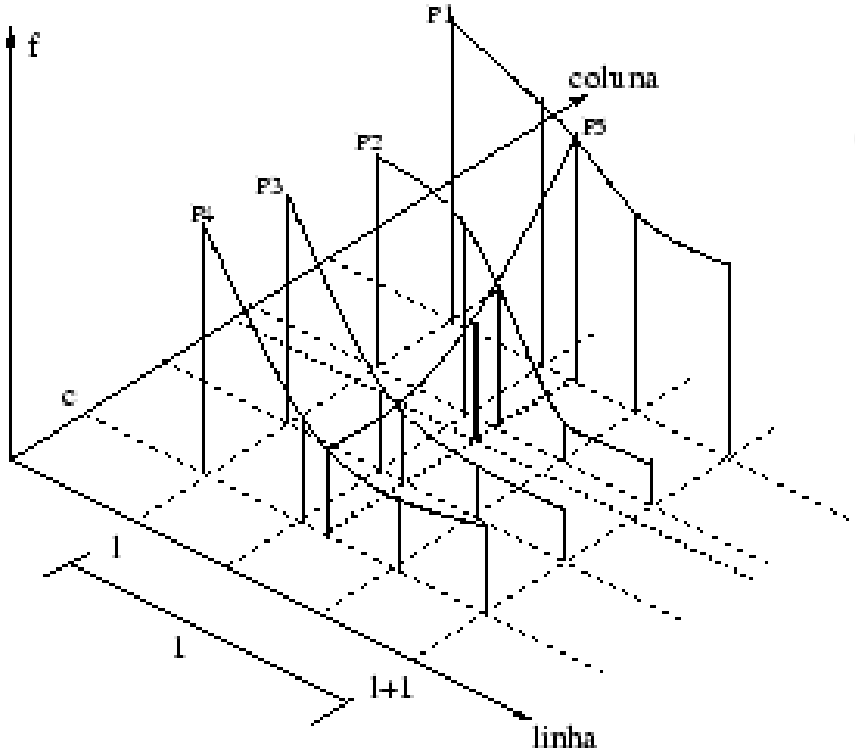


http://northstar-www.dartmouth.edu/doc/idl/html_6.2/Interpolation_Methods.html

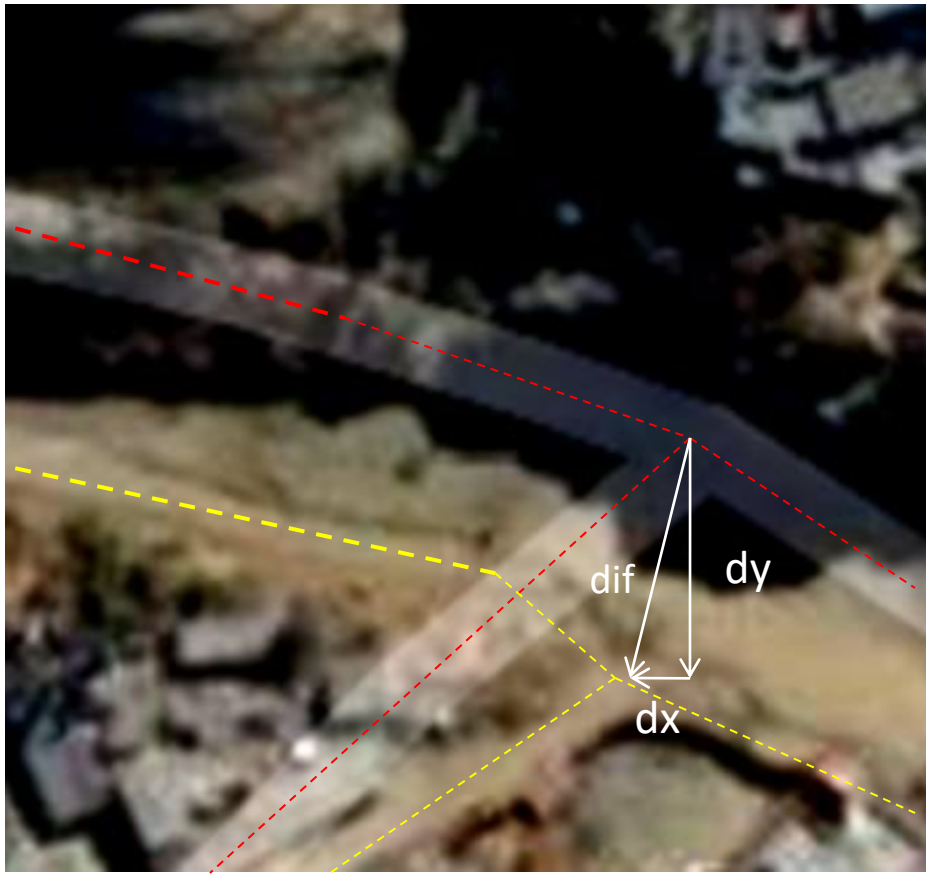
Convolução cúbica:

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





Qualidade do produto



- Com: $dif^2 = dx^2 + dy^2$

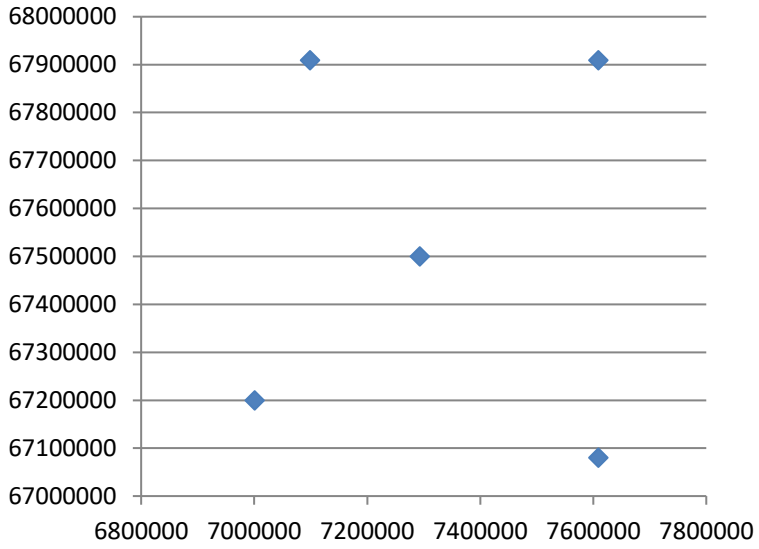
Avaliação da qualidade

- É feita de maneira similar à avaliação do modelo, mas adotando pontos de verificação, diferentes dos de controle usados para gerar o modelo.

- Medir coordenadas na imagem georreferenciada (nova) e comparar estas coordenadas com dados de campo (mapa).

- Atende à escala? O sensor é apropriado?

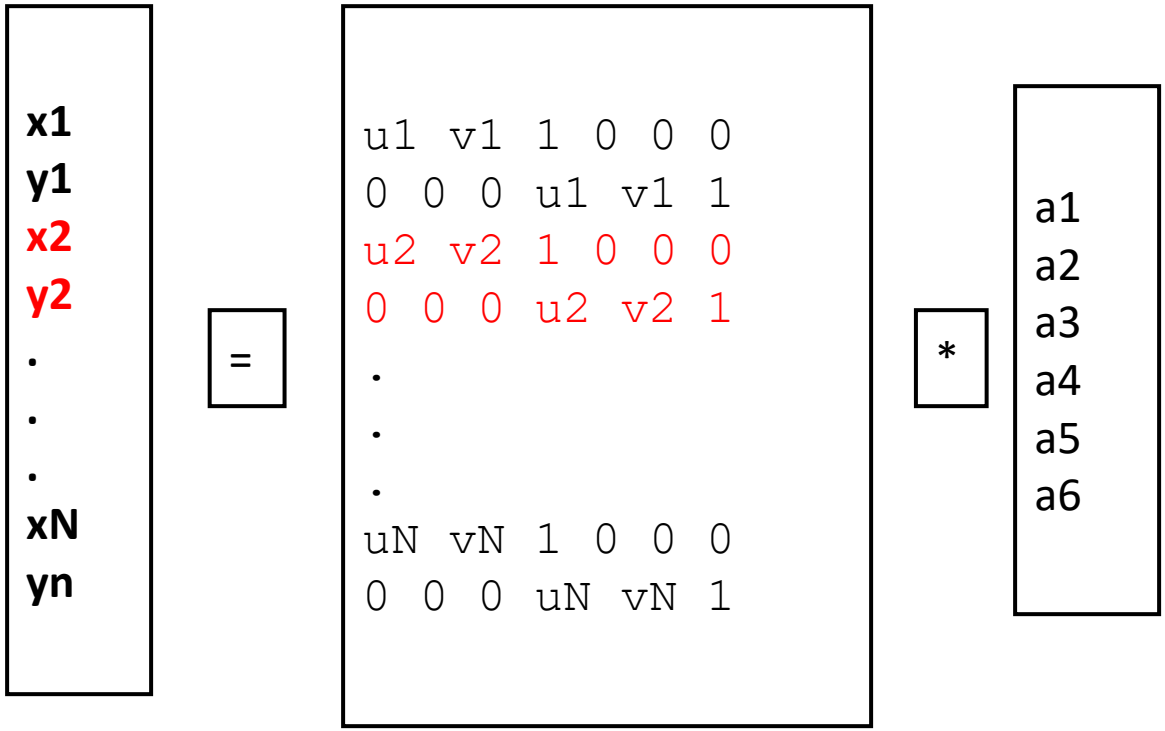
Erro médio= soma de erros/N



L	N	L1	N1	dL	DN	dist
7099071	67908765	7099102	67908755	31	-10	32,6
7609000	67080005	7609012	67080034	12	29	31,4
7293087	67500001	7293092	67500009	5	8	9,4
7000871	67200001	7000871	67200046	0	45	45,0
7609000	67908765	7609012	67908756	12	-9	15,0
					Média=	26,7

Erro médio= soma de erros/N

Ajustamento programa



Ou na forma matricial: $T = W * A$

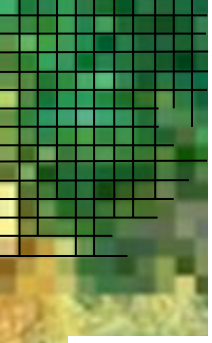
```
# coordenadas de pontos de apoio: X Y de imagem e X Y de terreno
P = np.array([ [ 81756 , 90767 ,  597 ,   180],
               [ 77258 , 78218 , 376.33 , 598.33],
               [ 69720 , 86446 , 135.67 , 314.33],
               [ 79996 , 78231 ,  450 ,   618.6],
               [ 67238 , 72769 , 35.67 ,  767.17],
               [ 78317 , 90357 , 426.67 , 195],
               [ 80989 , 84798 , 509 ,    383.2],
               [ 71148 , 75798 , 174.4 ,  663.8] ])
npt,nv=P.shape # tamanho da matriz npt=nro de pontos, nv=colunas
print('Num ptos=',npt, ' : ncolunas= ', nv)
```

```
## Separara, da matriz P, as coordenadas de campo e construir vetor
# em Y
Y0=[] ; # inicializar matriz, tamanho 2n (2 linhas por ponto
Y0 = [0 for i in range(npt*2)] # tudo zero no inicio
for i in range(npt): # Copia uma coordenada na linha par, outra na impar
        Y0[2*i ]=P[i,0]
        Y0[2*i+1]=P[i,1]
b = np.array([Y0]) # muda para um vetor (array) e tranpor usando funcao "T"
Y=b.T
print('Vetor de coord de mapa') #mostrar, pode apagar depois
print(Y)
## Construir matriz de coordenadas de tela com as coordenadas 2 e 3 da matriz P
X = np.zeros( ( 2*npt,6) ) # cria matriz com zeros com 2NPT linhas e 6 colunas
print(X)
for i in range(npt): # preenche a matriz
         X[2*i,0 ]=P[i,2]
         X[2*i,1 ]=P[i,3]
         X[2*i,2 ]=1
         X[2*i+1,3]=P[i,2]
         X[2*i+1,4]=P[i,3]
         X[2*i+1,5]=1
print('Matriz coord de imagem')
print(X)
```

```
## solucao em ajustamemnto
# Y= X * A
# X' * Y = X' * X * A
# A= inv( X' * X )* X' * Y

## transpor matriz X
Xt= np.transpose(X)
print('Matriz transposta')
print(Xt)
## produto vetorial= dot (dot product)
C=np.dot(Xt,X)
print(C.shape)
```

```
D=np.linalg.inv(C) # inverter matriz 6x6
# # se deu certo, deve dar matriz identidade
# I=np.dot(C,D)
# print(I)
## solucao em ajustamemnto
# Y= X * A
# X' * Y = X' * X * A
# A= inv( X' * X )* X' * Y
B=np.dot(Xt,Y)
A=np.dot(D,B)
print('solucao')
print(A)

## calcule residuos
R= np.dot(X,A)

Dif=R-Y
print('real, estimados, diferenca')
for i in range(npt):
# print(Y[2*i], R[2*i], Dif[2*i] , ' | ', Y[2*i+1], R[2*i+1], Dif[2*i+1])
print(": %5.2f %5.2f %5.2f %5.2f %5.2f %5.2f " % ( Y[2*i], R[2*i], Dif[2*i] , Y[2*i+1], R[2*i+1], Dif[2*i+1]
) )
```

: 81756.00 82776.06 **1020**.06 90767.00 90978.26 211.26
: 77258.00 77247.77 -10.23 78218.00 78315.86 97.86
: 69720.00 69466.42 -253.58 86446.00 86275.52 -170.48
: 79996.00 79471.96 -524.04 78231.00 77839.88 -391.12
: 67238.00 67584.40 346.40 72769.00 72794.99 25.99
: 78317.00 77780.04 -536.96 90357.00 90259.38 -97.62
: 80989.00 80657.57 -331.43 84798.00 84858.91 60.91
: 71148.00 71437.77 289.77 75798.00 76061.19 263.19



T = 1.0e+04 *

X	y	Xe	Ye	Dx	Dy	RMS
0	1.8237	0.0004	1.8240	-0.0004	-0.0003	0.0005
1.6288	1.7998	1.6268	1.8006	0.0020	-0.0008	0.0022
1.1790	0.5449	1.1796	0.5445	-0.0006	0.0004	0.0008
0.1770	0	0.1751	0.0002	0.0019	-0.0002	0.0020
0.5680	0.3029	0.5693	0.3028	-0.0013	0.0001	0.0013
1.5521	1.2029	1.5522	1.2040	-0.0001	-0.0011	0.0011
1.2693	1.3698	1.2709	1.3689	-0.0016	0.0009	0.0018
1.2849	1.7588	1.2848	1.7577	0.0001	0.0011	0.0011

Media=13m

$$PV=[L\ N\ x\ y]$$

$$PV=\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}$$

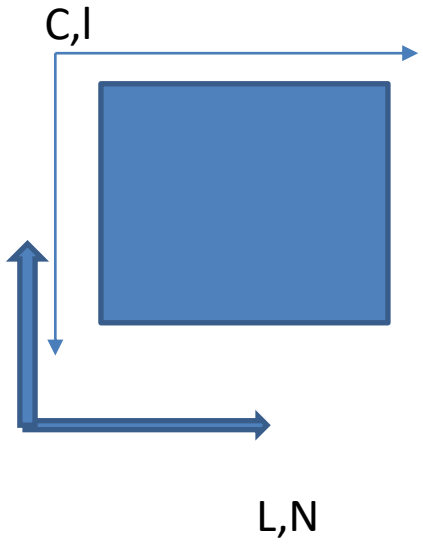
$$PP=PV*A$$

$$OBS=[L;N]$$

$$Dife=PP-OBS$$

programa

A) Calcule os valores dos parâmetros da transformação de 1ª ordem
B) A=[a0 a1 a2 z3 a4 a5]
C) Ler a imagem de tamanho Nlin, Ncol
D) Delimitar a área que será corrigida, em coordenadas Leste, Norte (pode fazer isto com base nos pontos de apoio) `Lmin, Lmax, Nmin, Nmax`
E) Com o valor da resolução da imagem `resol`
F) Calcule o numero de linhas e colunas
   A) `Nlin=round( (Nmax-Nmin)/resol) +1`
   B) `Ncol=round( (Emax-Emin)/resol) +1`
G) E gere uma matriz vazia desse tamanho




```
Varrer a imagem de saída
For lin in range(Nlin)
        For col in range(Ncol)
                Transformar a L,N
                L= col*resol + Lmin
                N=lin*resol - Nmax
                calcule lin0/col0 da imagem original
                col0=a0*L + a1*N + a2
                 lin0=a3*L + a4*N + a5
                if col0>0 & col0<Ncol0        & lin0>0 & lin0<Nlin0
                        # 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(L,C)  *(1-DL) +   I(L+1,C) *DL
H2=I(L,C+1)*(1-DL) + I(L+1,C+1) *DL
H=H1*(1-DC) + H2*DC
```