Processamento de Imagens

Reamostragem de Imagens

Prof. Linder Cândido da Silva

UFMT

Julho de 2024

Agenda

- **Interpolação de imagens.**
- Reamostragem por interpolação baseada no vizinho mais próximo.
- Reamostragem por interpolação bilinear.
- cv2.resize() do pacote OpevCV.

Introdução

- **Reamostrar** significa aumentar ou diminuir o número de pixels de uma imagem para fins de ampliação ou redução.

- Para isso, usamos técnicas de **interpolação**. Ou seja, usamos dados conhecidos para estimar valores em localizações desconhecidas desconhecidas da imagem ampliada ou reduzida.

- Os principais métodos de interpolação usados em processamento de imagens são a interpolação **por vizinho mais próximo**, **bilinear**, e bicúbica.

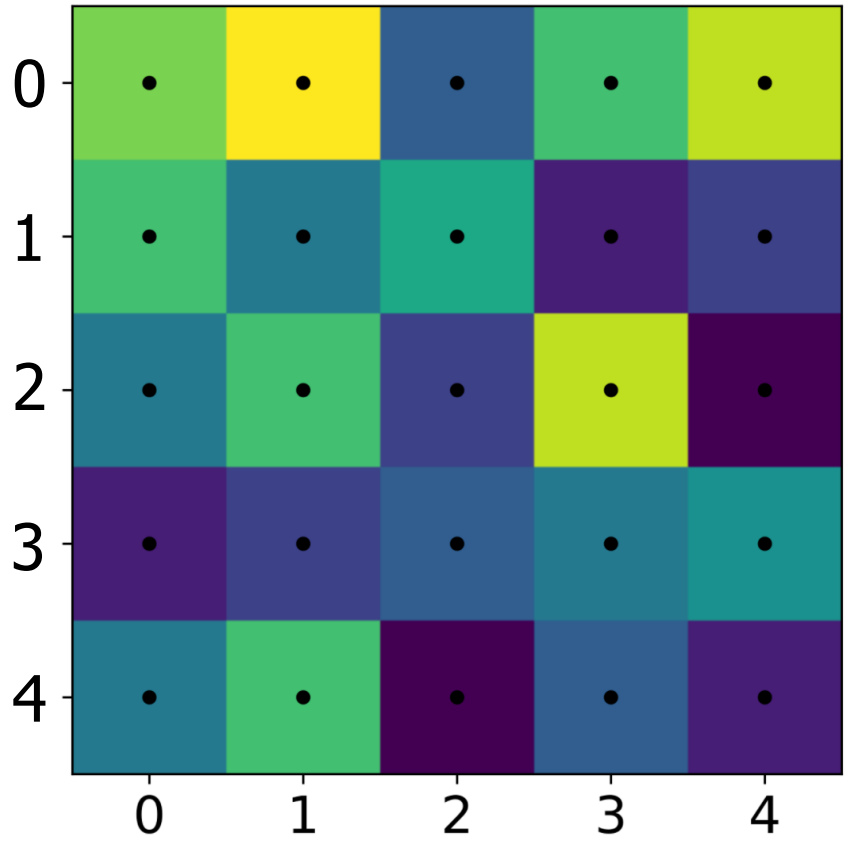
- Existem inúmeras aplicações com imagens que utilizam métodos de interpolação. Por exemplo, zoom, redução e correções em transformações geométricas.

Agenda

- Interpolação de imagens.
- **Reamostragem por interpolação baseada no vizinho mais próximo.**
- Reamostragem por interpolação bilinear.
- cv2.resize() do pacote OpevCV.

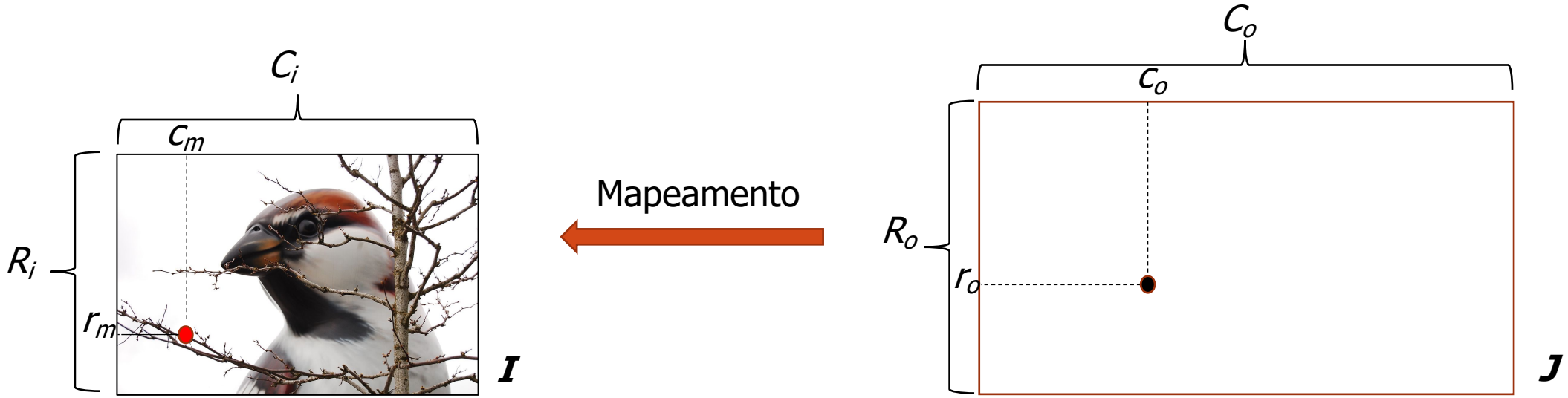
Reamostragem por Vizinho Mais Próximo

- Neste método, o valor estimado é sempre igual ao da sua amostra mais próxima, desconsiderando qualquer outra.
- Método simples regularmente utilizado para interpolações rápidas e em áreas de estudo envolvendo dados (imagens) bem amostrados.
- Na figura ilustrativa ao lado, cada célula colorida indica a área de influência dos pontos pretos quando a interpolação por vizinho mais próxima é aplicada.



Reamostragem por Vizinho Mais Próximo

- Dada a imagem original I, com dimensões $R_i \times C_i$, e a imagem reamostrada J, com dimensões $R_o \times C_o$. Quando J é maior que I, temos uma ampliação (zoom). Quando J é menor que I, temos uma redução.
- Procedimento:
  1. Mapeamos pixels com coordenadas (r_o, c_o) de J para posições (r_m, c_m) de I.
  2. Atribuímos a cada pixel (r_o, c_o) de J o valor do pixel mais próximo a (r_m, c_m) em I.

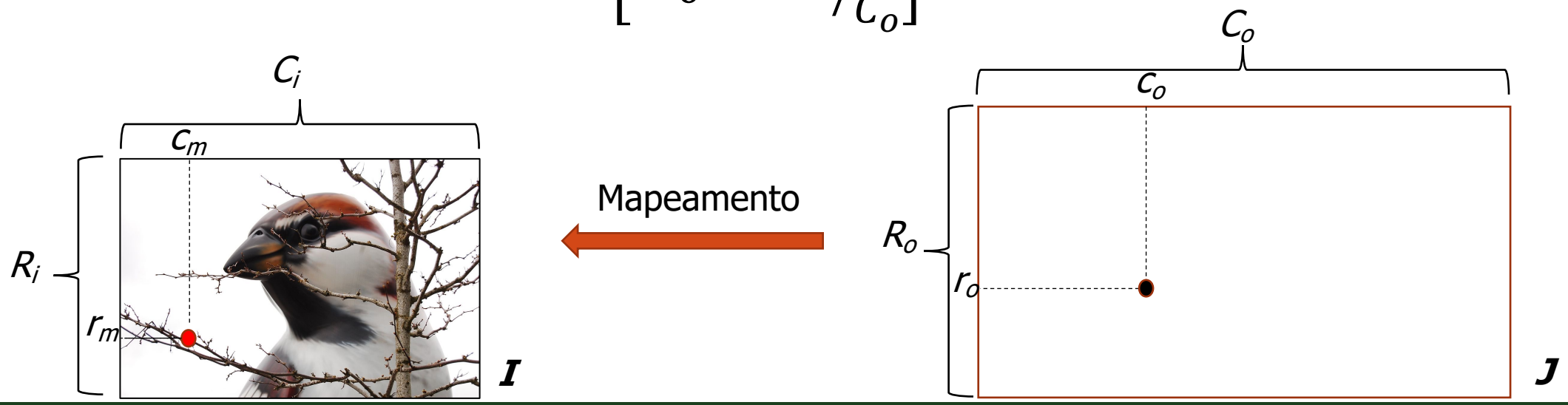


Mapeamento das Posições dos Pixels de J

- A posição (r_m, c_m) em I é obtida aplicando uma transformação de escala nas coordenadas (r_0, c_o), tal como faríamos se quiséssemos transformar as dimensões de J para que ficassem iguais às dimensões da imagem de entrada I.

- <u>Matematicamente</u>:

$$\begin{bmatrix} r_m \\ c_m \end{bmatrix} = \begin{bmatrix} R_i/R_o & 0 \\ 0 & C_i/C_o \end{bmatrix} \begin{bmatrix} r_o \\ c_o \end{bmatrix}$$



Reamostragem por Vizinho Mais Próximo

- **Algoritmo**:
  1. Dados:
     - Imagem original **I**: $R_i \times C_i$
     - Imagem de saída **J**: $R_o \times C_o$
  2. Fator de escala:
     - Na linha S_r: R_i / R_o
     - Na coluna S_c: C_i / C_o
  3. Mapeamento:
     - $\mathbf{r_m} = [0\ 1\ 2\ \ldots\ R_o\text{-}1] \times S_r$
     - $\mathbf{C_m} = [0\ 1\ 2\ \ldots\ C_o\text{-}1] \times S_c$
  4. Coord. dos Viz. mais próximos:

     $\mathbf{r} = \text{round}(\mathbf{r_m})$ e $\mathbf{c} = \text{round}(\mathbf{c_m})$

  5. Para l=0..R_o-1, m=0..C_o-1:
     - **J**(l,m) = **I**(**r**[l],**c**[m])

I

110	55	233	77
122	18	143	215
11	10	255	72
48	137	135	87

J

Efeitos Indesejáveis

- A reamostragem por vizinho mais próximo é um método simples para redimensionar imagens, mais apresenta alguns efeitos indesejáveis:
  - A cópia direta dos valores dos pixels pode resultar em transições irregulares com "dentes de serra" (**aliasing**), especialmente em linhas diagonais ou curvas.
  - **Perda de detalhes finos** da imagem original, particularmente nas reduções.
  - Em ampliações, a imagem pode parecer **pixalizada**, já que os mesmos pixels são replicados para preencher o espaço.
  - Podem ocorrer **artefatos visuais** em torno de bordas ou áreas de contraste, tornando as transições entre cores ou intensidades abruptas e menos naturais.
- Esses efeitos indesejáveis tornam a reamostragem por vizinho mais próximo menos usada em aplicações onde a qualidade visual é crucial, como em fotografia ou design gráfico.

Exemplo

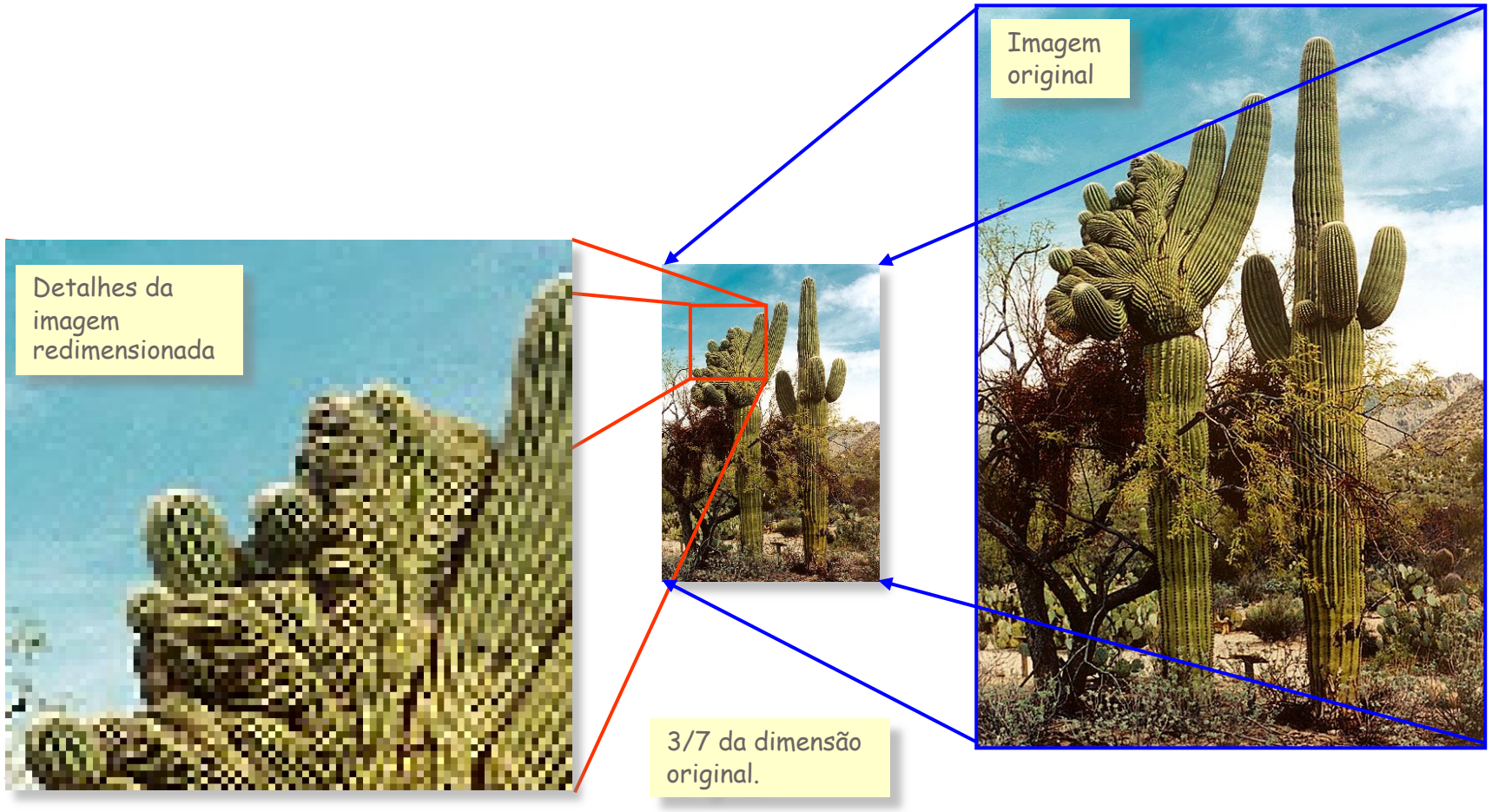


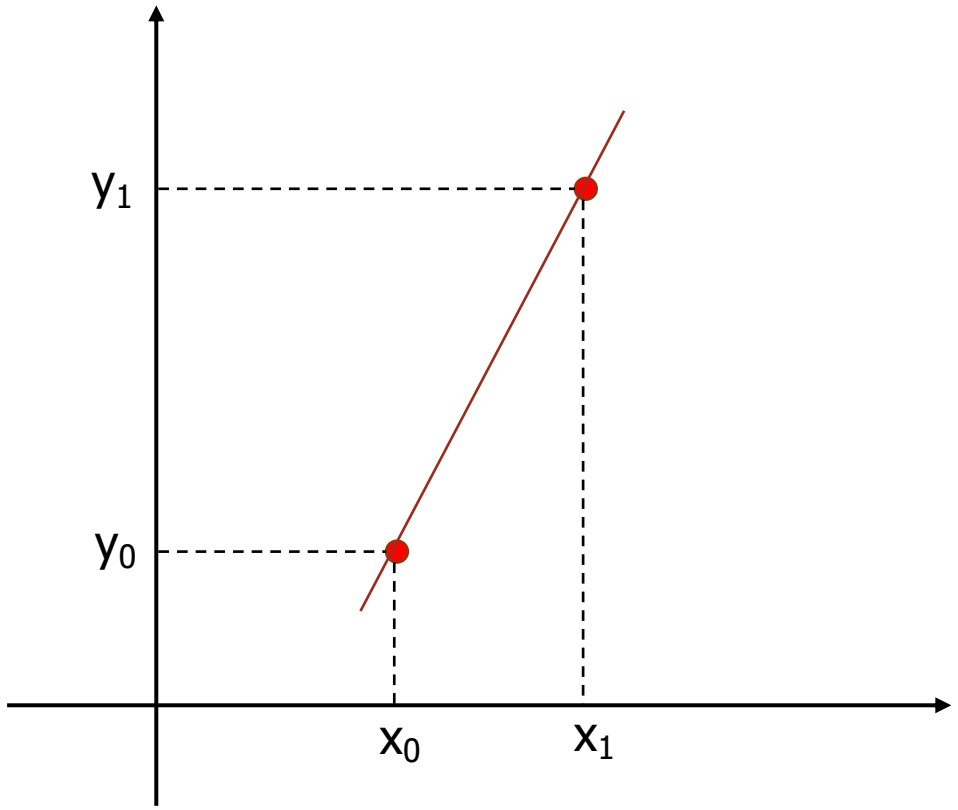
Imagem original
Detalhes da imagem redimensionada
3/7 da dimensão original.

Agenda

- Interpolação de imagens.
- Reamostragem por interpolação baseada no vizinho mais próximo.
- **Reamostragem por interpolação bilinear.**
- cv2.resize() do pacote OpevCV.

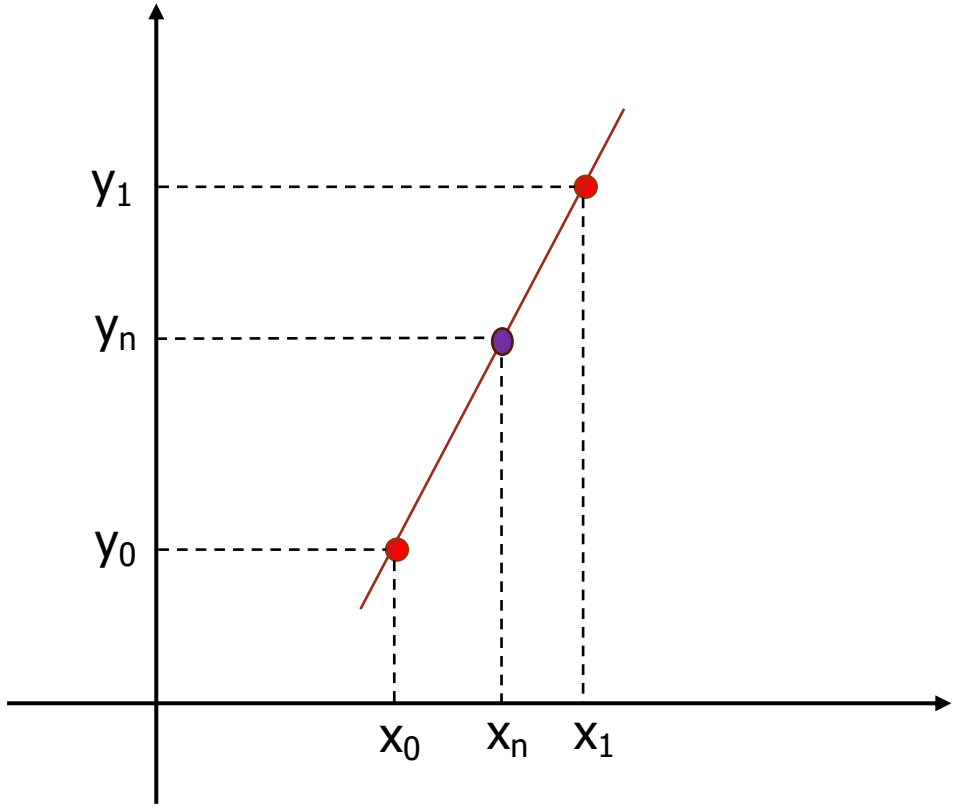
Revisão: Interpolação Linear

- Dados dois pontos conhecidos (x_0, y_0) e (x_1, y_1), obtemos o polinômio interpolador de grau 1 traçando uma reta entre eles.



Revisão: Interpolação Linear

- Dada um novo valor x_n obtemos y_n de acordo com a equação da reta ligando os dois pontos conhecidos. Basicamente, a estimativa é uma média ponderada dos valores conhecidos.



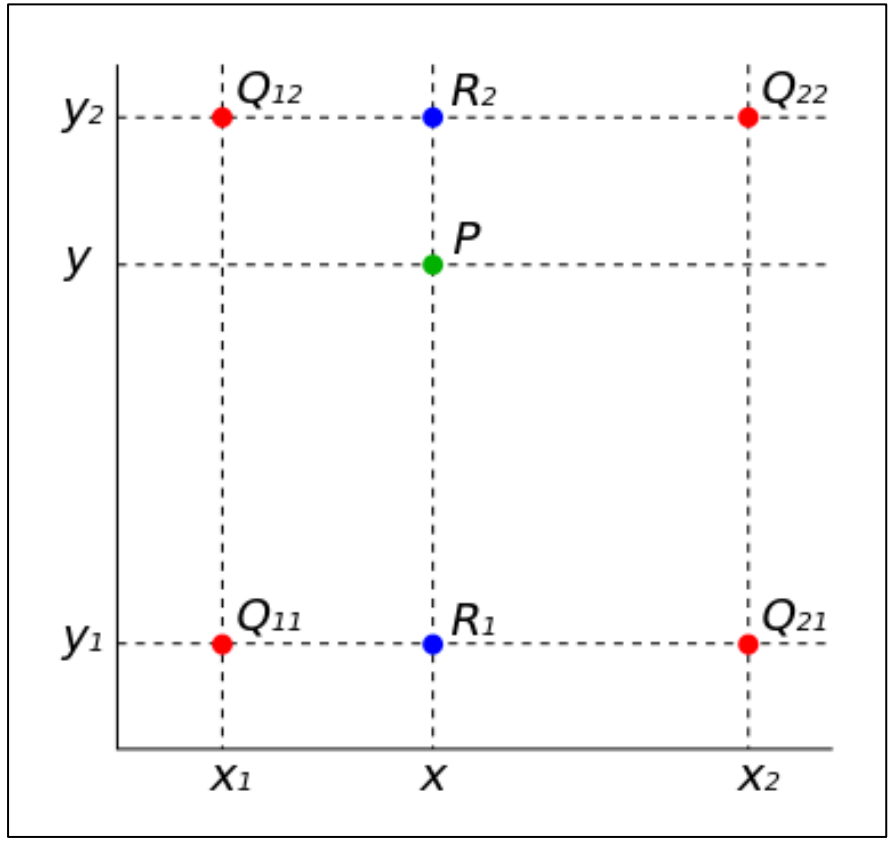
$$y_n - y_0 = \left(\frac{y_1 - y_0}{x_1 - x_0}\right)(x_n - x_0)$$

...

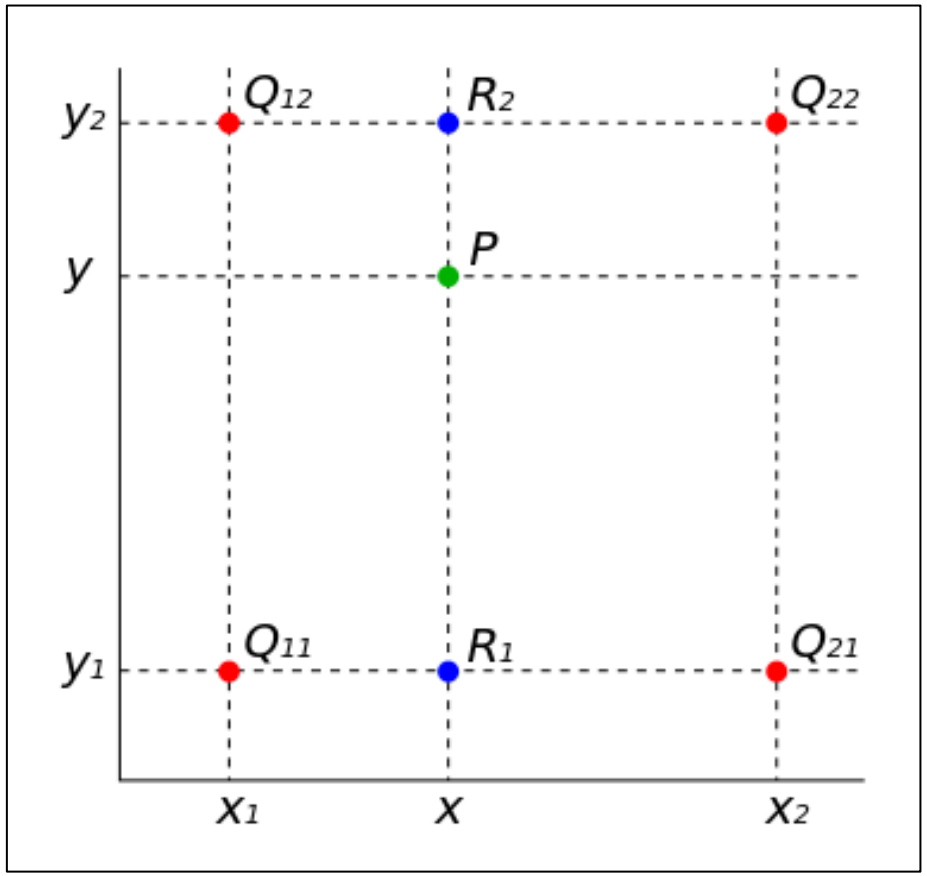
$$y_n = \frac{y_1(x_n - x_0)}{x_1 - x_0} + \frac{y_0(x_1 - x_n)}{x_1 - x_0}$$

Interpolação Bilinear

- A reamostragem bilinear é uma extensão da interpolação linear para funções de duas variáveis em um reticulado 2d.
- Suponha que queiramos encontrar o valor da função desconhecida f no ponto $p = (x, y)$. Para isso, usamos os quatro pontos $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$ e $Q_{22} = (x_2, y_2)$.
- Em seguida, realizamos interpolações lineares primeiro em uma direção e depois novamente na direção perpendicular.



Interpolação Bilinear



Primeiro a interpolação linear na direção x

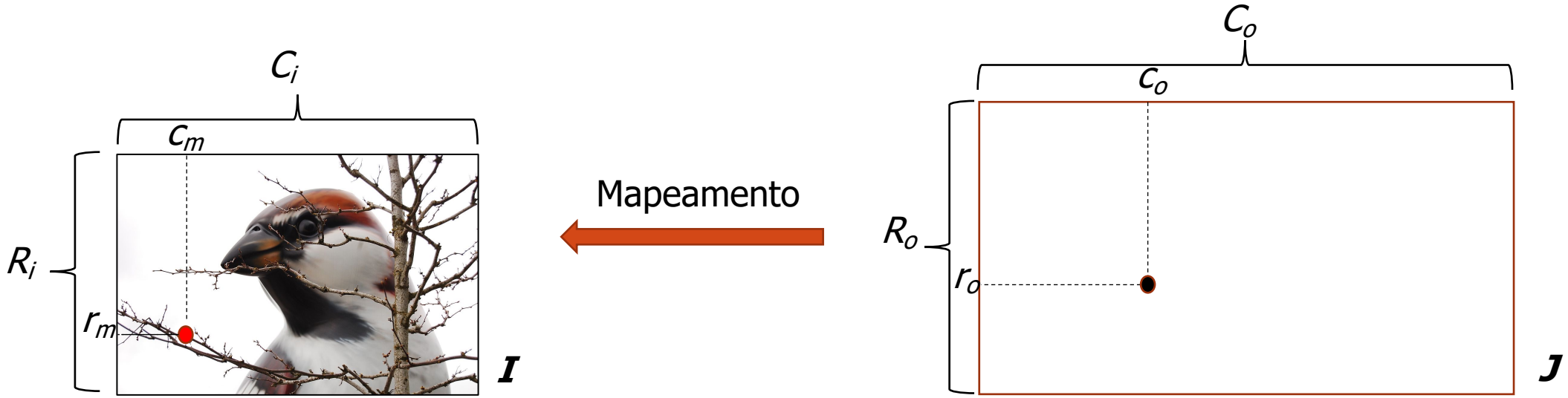
$$R_1 = f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$
$$R_2 = f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

Depois a interpolação linear na direção y

$$P = f(x, y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$

Reamostragem por Interpolação bilinear

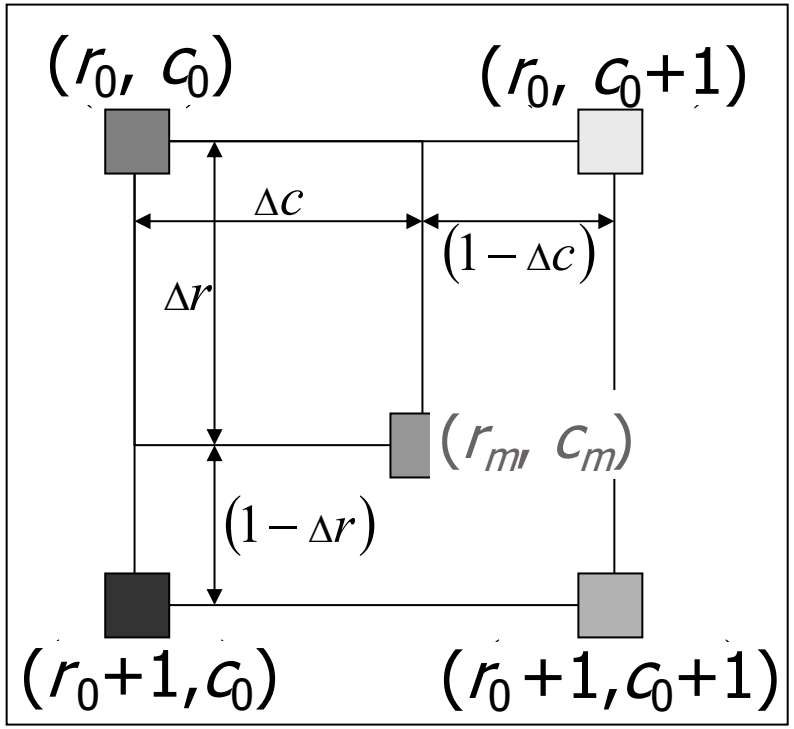
- Tal como no caso da interpolação por vizinho mais próximo, mapeamos as posições da imagem de saída J para posições na imagem de entrada I: $(r_m, c_m) = ((R_i/R_o)r_o\,, (C_i/C_o)c_o)$
  - Posições (r_o, c_o) de J são mapeadas para as posições (r_m, c_m) na imagem de entrada I.
  - Atribuímos a cada pixel (r_o, c_o) de J o valor calculado com interpolação bilinear no ponto (r_m, c_m) de I.



Passo-a-Passo

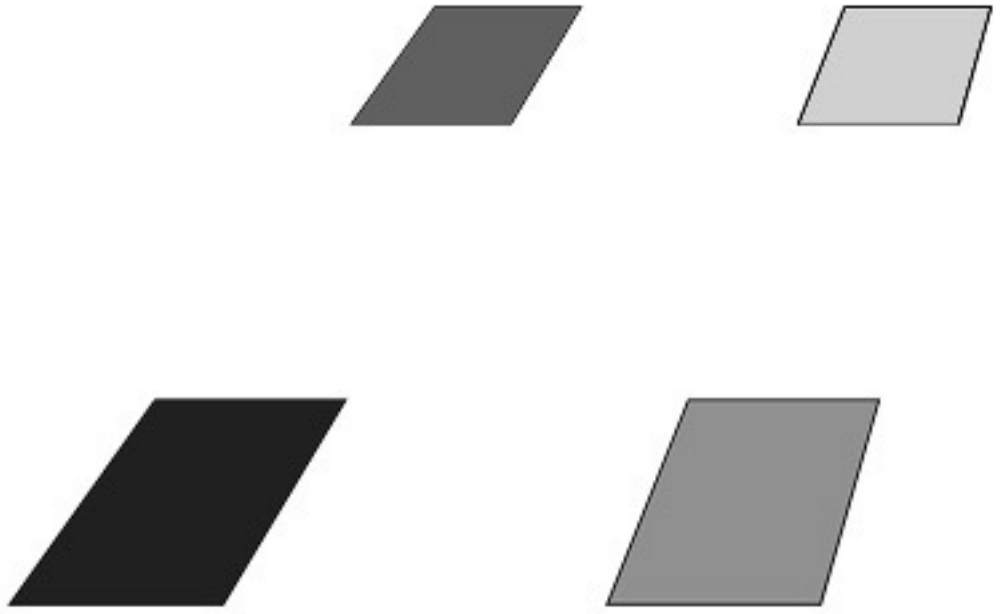
- Para cada pixel com coordenadas (r, c) da imagem de saída J, faça o seguinte:

1. Mapeie (r, c) para (r_m, c_m) na imagem original I.
2. Determine os 4 vizinhos mais próximos de (r_m, c_m): $\{(r_0, c_0), (r_0, c_0+1), (r_0+1, c_0), (r_0+1, c_0+1)\}$.
3. Faça uma interpolação linear no ponto (r_0, c_m) usando $\{(r_0, c_0), (r_0, c_0+1)\}$.
4. Faça uma interpolação linear no ponto $(r_0 + 1, c_m)$ usando $\{(r_0+1, c_0), (r_0+1, c_0+1)\}$.
5. Finalmente, faça uma interpolação linear no ponto (r_m, c_m) usando $\{(r_0, c_m), (r_0 + 1, c_m)\}$.
6. Atribua ao pixel (r, c) de J o valor interpolado em I na posição (r_m, c_m).



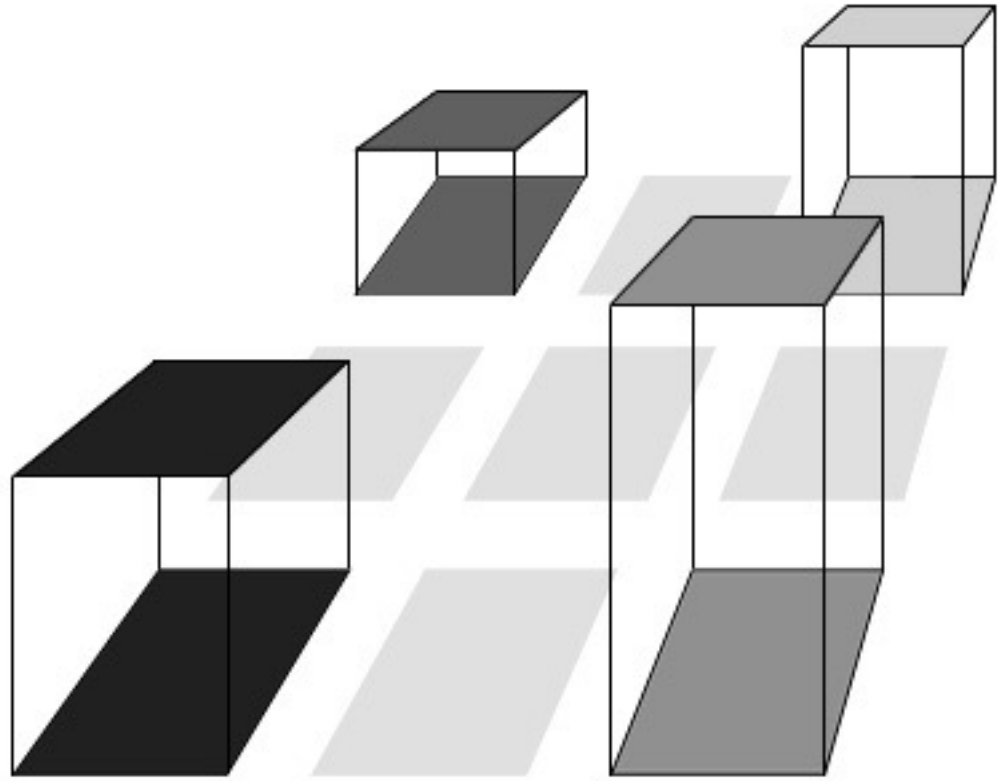
Ilustração

Seguindo a ideia que apresentamos nos slides anteriores. Queremos aproximar um ponto entre os quatro pixels destacados. Os tons de cinza ilustram o valor da função e devem ser imaginados como alturas acima do plano da imagem.



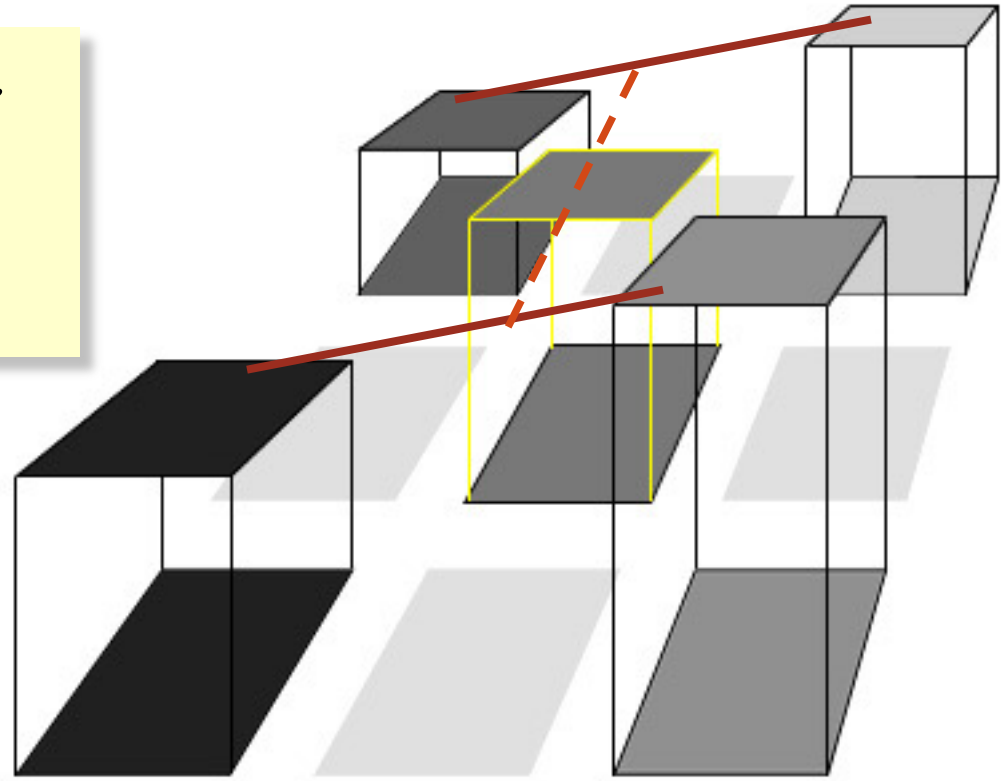
Ilustração

Em cinza claro destacamos os pixels cujos valores podem ser inferidos por interpolação bilinear.



Ilustração

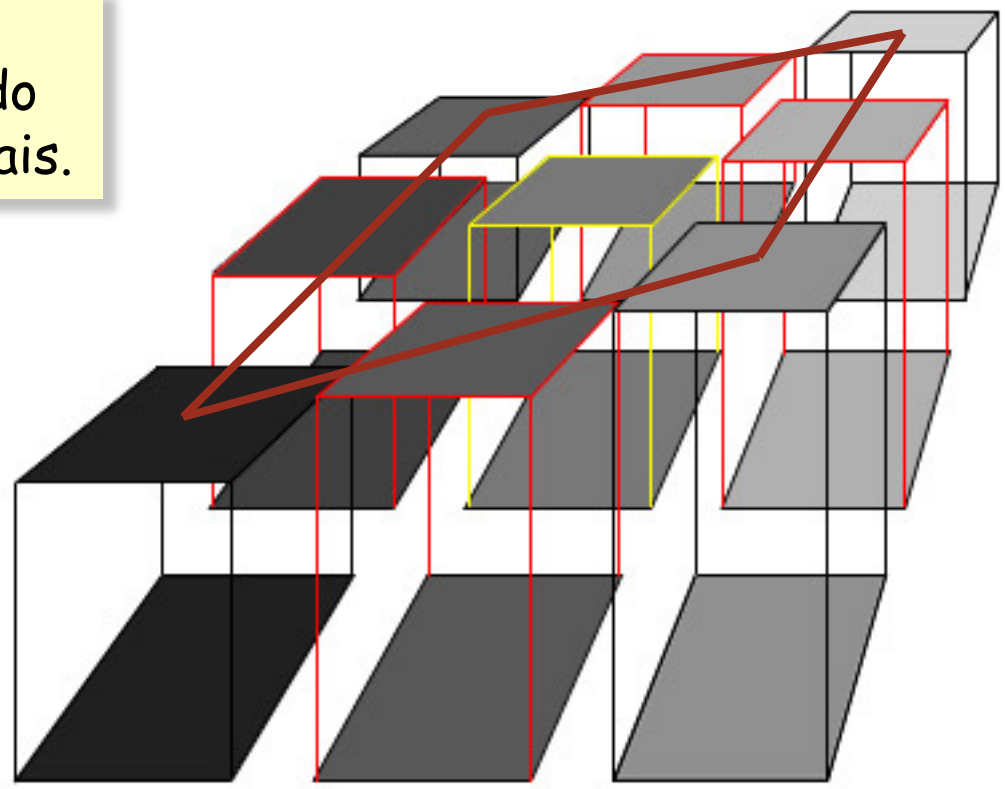
Imagine os níveis de cinza como alturas acima do plano da imagem.



Centro = média ponderada das quatro alturas.

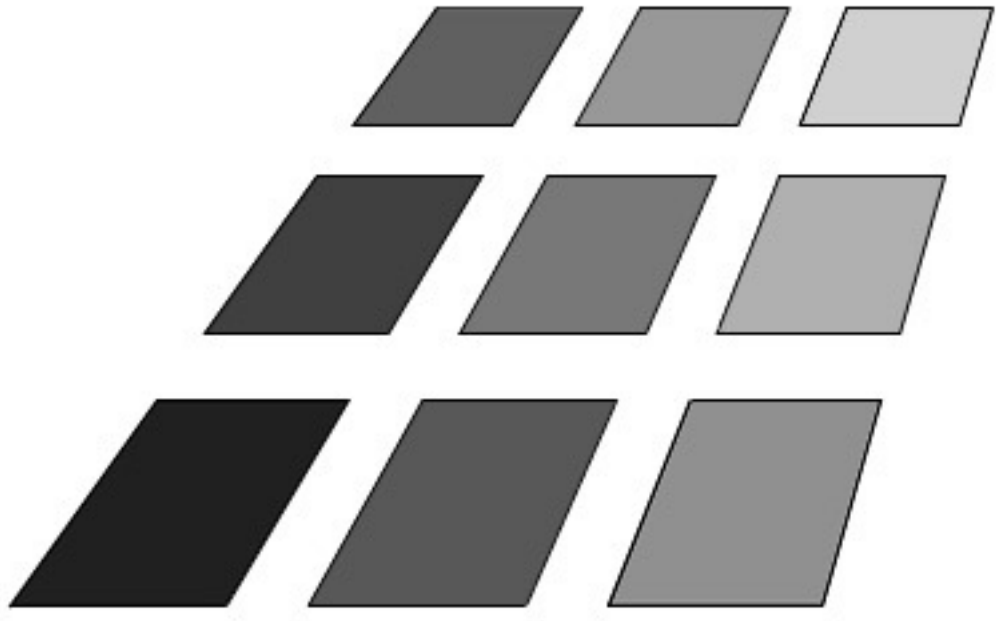
Ilustração

Valores dos novos pixels caem na retas conectando valores dos pixels originais.



Ilustração

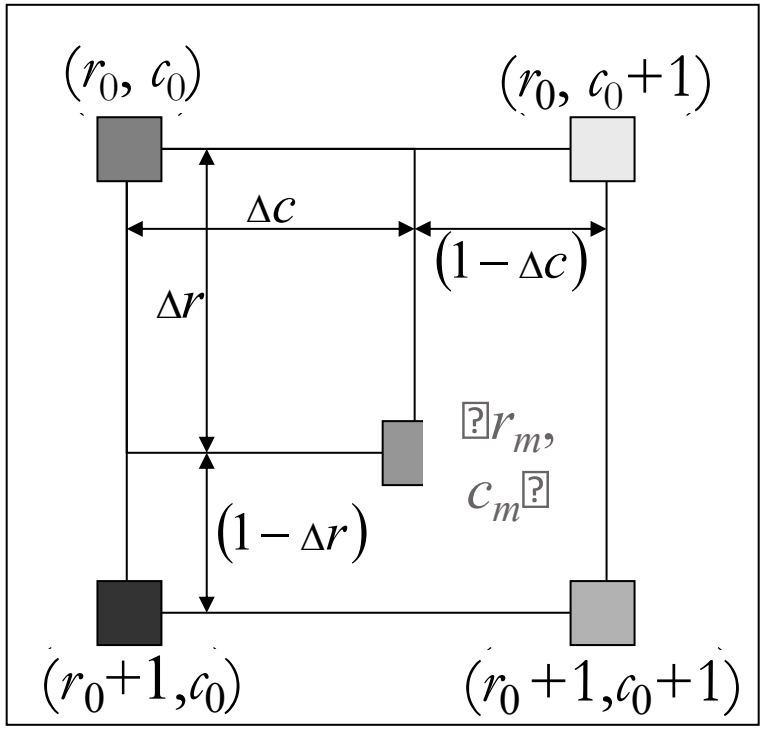
O resultado final:



Reamostragem Bilinear

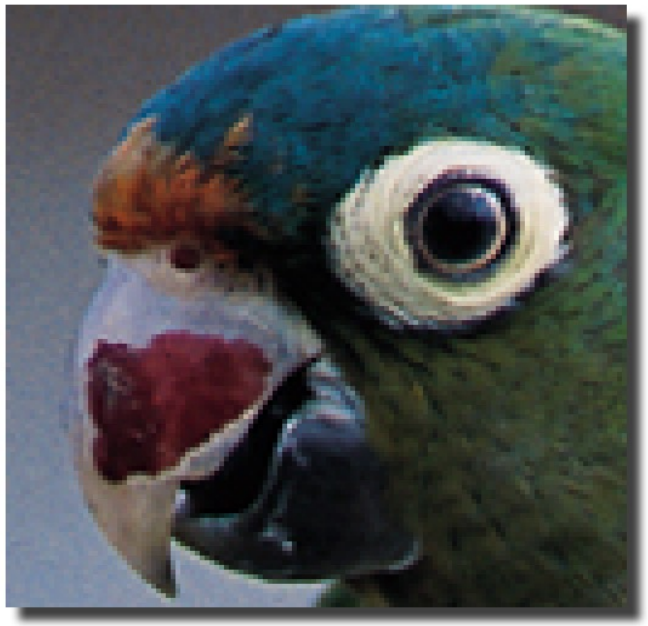
- Algoritmo:
  1. Dados:
     - Imagem original **I**: $R_i \times C_i$
     - Imagem escalada **J**: $R_o \times C_o$
  2. Fator de escala:
     - S_r: R_i/R_o e S_c: C_i/C_o
  3. Mapeamento:
     - $\boldsymbol{r_m} = \boldsymbol{r} \times S_r$ para $\boldsymbol{r}$=[0 1 2 ... R_o-1]
     - $\boldsymbol{C_m} = \boldsymbol{c} \times S_c$ para $\boldsymbol{c}$=[0 1 2 ... C_o-1]
  4. Calcule (r_0, c_0) para cada par (r_m, c_m):
     $$(r_0, c_0) = (\text{floor}(r_m), \text{floor}(c_m))$$
  5. Para cada par (r_m, c_m) obtido de (r,c):

$$\begin{aligned}\mathbf{J}(r,c) = {} & \mathbf{I}(r_0, c_0)\cdot(1-\Delta r)\cdot(1-\Delta c) \\ & + \mathbf{I}(r_0+1, c_0)\cdot\Delta r\cdot(1-\Delta c) \\ & + \mathbf{I}(r_0, c_0+1)\cdot(1-\Delta r)\cdot\Delta c \\ & + \mathbf{I}(r_0+1, c_0+1)\cdot\Delta r\cdot\Delta c.\end{aligned}$$



Reamostragem Bilinear

Viz. mais próximo



Bilinear

Melhor qualidade visual!

Agenda

- Interpolação de imagens.
- Reamostragem por interpolação baseada no vizinho mais próximo.
- Reamostragem por interpolação bilinear.
- **cv2.resize() do pacote OpevCV.**

Opencv `resize()`

- Opencv disponibiliza a função `cv2.resize()` para a realização de reamostragem de imagens:
- `cv2.resize(src, dsize[, dst[, fx[, fy[, interpolation]]]])`

Parâmetros	**Descrição**
src	[requerido] imagem de entrada.
dsize	[requerido] tamanho desejado da imagem de saída.
fx	[opcional] fator de escala horizontal
fy	[opcional] fator de escala vertical
interpolation	[opcional] flag: INTER_NEAREST (viz. mais próximo), INTER_LINEAR (bilinear), INTER_AREA (replicação de pixel), INTER_CUBIC (bicúbico).

Exemplo

```
1.  import numpy as np
2.  import cv2 as cv
3.  img = cv.imread('messi5.jpg')
4.  res = cv.resize(img,None,fx=2, fy=2, interpolation = cv.INTER_CUBIC)
5.  #OU
6.  height, width = img.shape[:2]
7.  res = cv.resize(img,(2*width, 2*height), interpolation = cv.INTER_CUBIC)
```