Processamento Digital de Imagens

Prof. Bogdan Tomoyuki Nassu

Hoje

- Redimensionamento de imagens.

Problema

- Onde este problema aparece?

Problema

- Onde este problema aparece?
  - Em **MUITOS** lugares!!!
- Alguns exemplos óbvios:
  - Visualização em programas de computador.
  - Dispositivos como televisores, celulares e video games.
  - Criação de *thumbnails*.

Outros usos

•Redimensionamento de imagens é muito usado para reduzir tempo de processamento ou espaço de armazenamento.

•Reduzir a imagem é o método mais simples de compressão.

•Em *games*, é muito comum renderizar certos elementos com uma resolução mais baixa, ampliando para compor a cena.

- Sombras.
- Reflexos.
- *Bloom*.
- Fundo borrado (*depth-of-field* ou *motion blur*).
- **TUDO** (*dynamic resolution scaling*, xbox one x ps4).

Sombras em baixa resolução

1080p

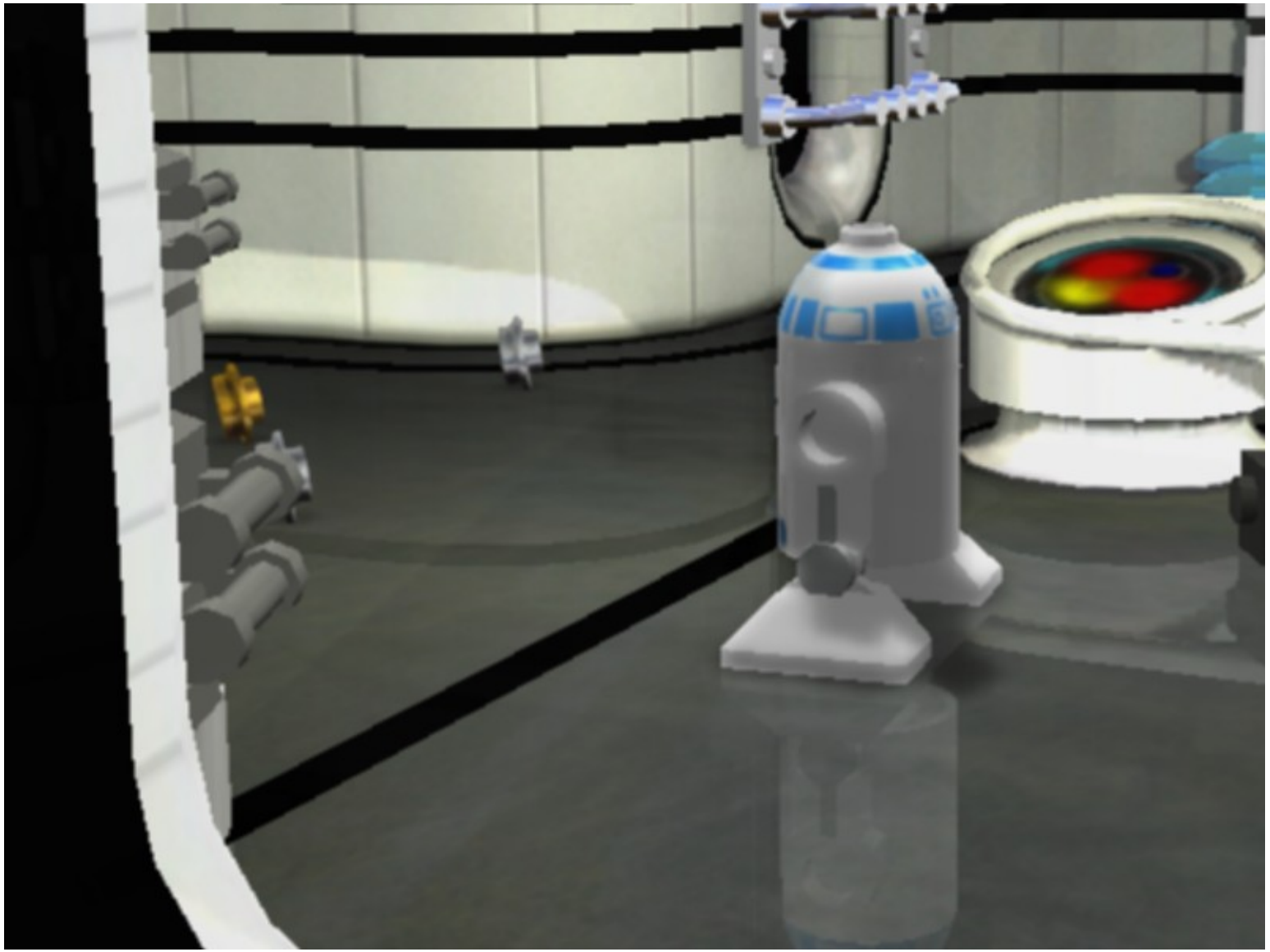
900p (ampliado para 1080p)

720p (ampliado para 1080p)

1080p

900p (ampliado para 1080p)

720p (ampliado para 1080p)



Digitalização

- Imagens digitais são obtidas por *amostragem* e *quantização*.

Digitalização

- Imagens digitais são obtidas por *amostragem* e *quantização*.
  - = discretizar um sinal contínuo.
- Amostragem: qual a distância (física) entre dois pixels?
  - Relação com resolução.
- Quantização: como o valor de um pixel é limitado?
  - Relação com número de bits por amostra.

Resoluções

Padrão / dispositivo	Resolução
VGA (máximo) / SD	640 x 480
DVD	720 x 480
720p (HD)	1280 x 720
1080p (Full HD)	1920 x 1080
4k (Ultra HD)	3840 x 2160
iPhone	320 x 480
iPhone 4	640 x 960
iPhone 6, 7, 8	750 x 1334
iPhone X	1125 x 2436
Game Boy	160 x 144
Game Boy Advance	240 x 160
Nintendo DS (cada tela)	256 x 192
Playstation Portable	480 x 272
Playstation Vita	960 x 544

Um exemplo simples

- Jogos de PSP podem ser executados pelo Playstation Vita.
  - Vita: 960x544 pixels.
  - PSP: 480x272 pixels.
  - *Upscaling*: imagem gerada com 480x272 pixels e redimensionada.

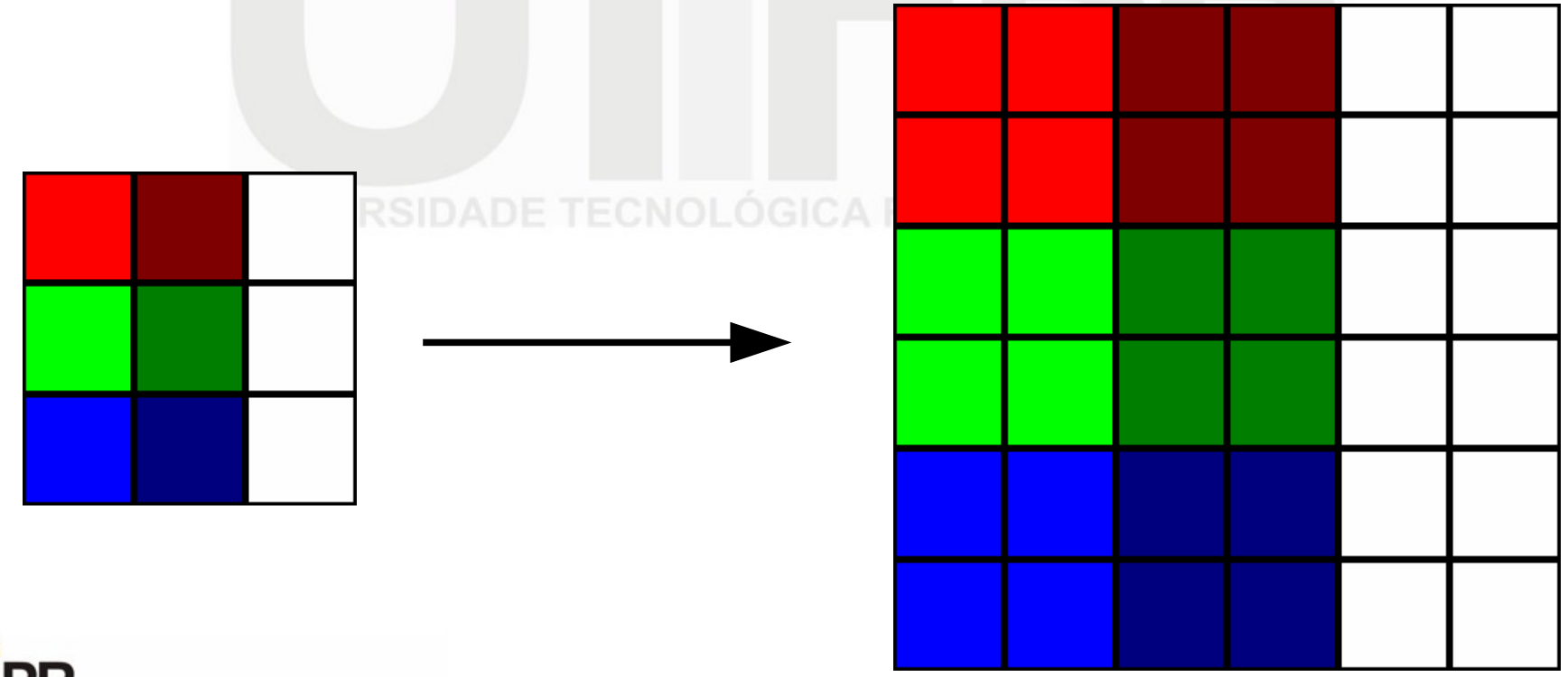
Como fazer?

- Como transformar uma imagem com 480x272 pixels em outra com 960x544 pixels?

Como fazer?

•Como transformar uma imagem com 480x272 pixels em outra com 960x544 pixels?

- Fator de escala de 2x (largura e altura).
- Basta duplicar cada pixel em cada direção!
- O número de pixels é quadruplicado.

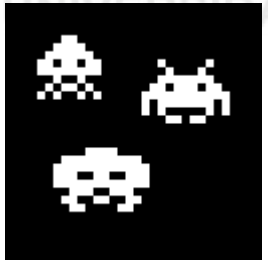
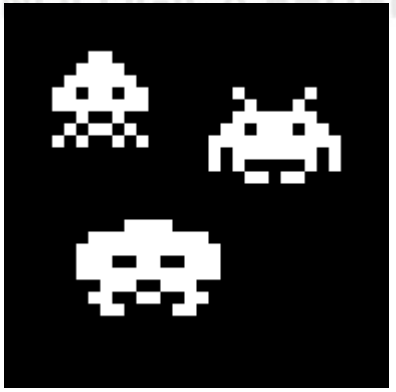
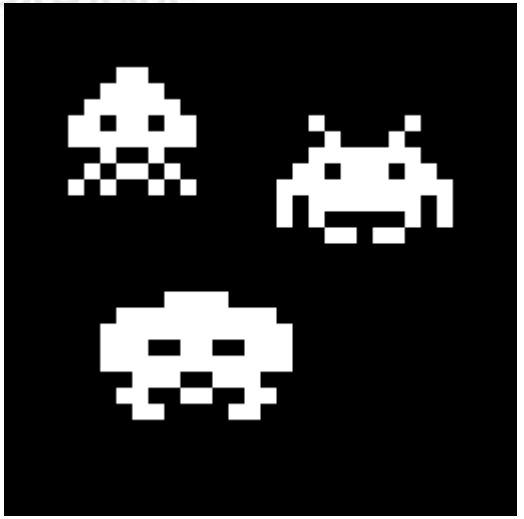


Ampliação, NN, fator inteiro positivo

- No exemplo, usamos uma abordagem de vizinho mais próximo (*nearest neighbor* / NN) e um único fator de escala inteiro positivo.
  - Para obter o valor do pixel em uma posição (x,y) da saída:
    - Dividimos x e y pelo fator de escala s e truncamos o resultado.
    - $g(x,y) = f(\text{floor}(x/s),\text{floor}(y/s))$
- O princípio é idêntico para qualquer fator de escala inteiro.
- Quais as vantagens desta abordagem?
- Quais as desvantagens desta abordagem?

Ampliação, NN, fator inteiro positivo

- Quais as vantagens desta abordagem?
  - Implementação rápida e simples.
  - O aspecto da imagem final é fiel à imagem original.
    - Esta propriedade é importante: a maior parte das imagens apresentadas nas aulas anteriores foi redimensionada desta forma.
- Quais as desvantagens desta abordagem?
  - Algumas imagens podem ficar com aspecto “pixelado”.
  - O que fazer se o fator de escala desejado não for inteiro?

Ampliação, NN

•Um caso mais geral: podemos usar a abordagem NN para fatores de escala não-inteiros.

- O que acontece neste caso?

Ampliação, NN

- Exemplo: emulador de Game Boy Advance em um PSP.
  - GBA: 240x160 pixels.
  - PSP: 480x272 pixels.
  - Fator de escala horizontal: 2
  - Fator de escala vertical: 1.7

Aspecto original...

Aspecto modificado.

Ampliação, NN

- Exemplo: emulador de Game Boy Advance em um PSP.
  - GBA: 240x160 pixels.
  - PSP: 480x272 pixels.
  - Fator de escala horizontal: 2
  - Fator de escala vertical: 1.7
- *Aspect ratio* (razão de aspecto?):
  - PSP: 30:17 (1.7647)
  - GBA: 3:2 (1.5)
- Como ampliar mantendo o *aspect ratio* original?

Ampliação, NN

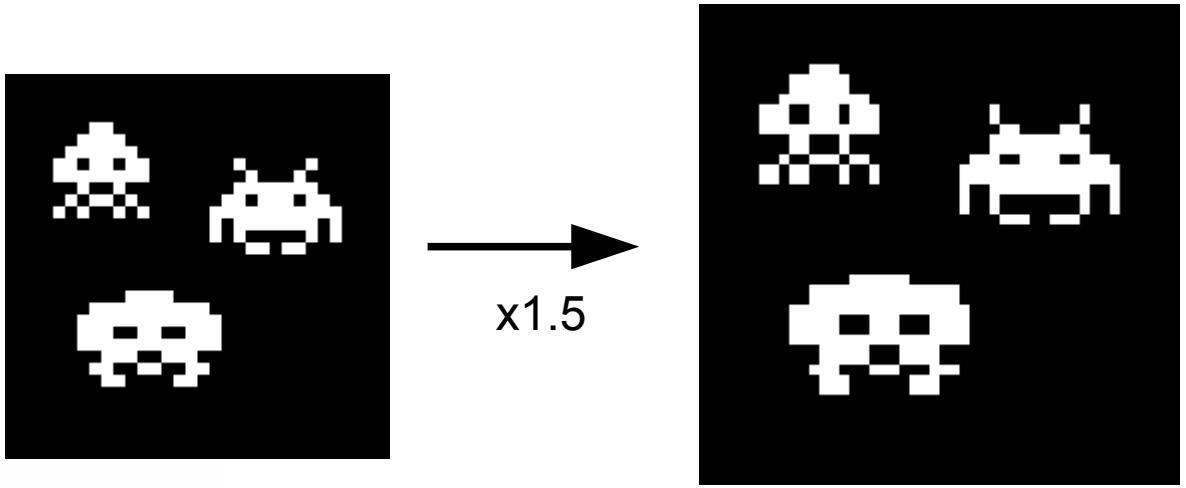
- Exemplo: emulador de Game Boy Advance em um PSP.
  - GBA: 240x160 pixels.
  - PSP: 480x272 pixels.
  - Fator de escala horizontal: 2
  - Fator de escala vertical: 1.7
- *Aspect ratio* (razão de aspecto?):
  - PSP: 30:17 (1.7647)
  - GBA: 3:2 (1.5)
- Como ampliar mantendo o *aspect ratio* original?
  - Basta usar o menor fator de escala para largura e altura.

Aspecto modificado.

Ampliação, NN

- Problema que pode surgir ao usarmos fatores não-inteiros: alteração na proporção de pixels.
- Exemplo (1D): coordenadas com fator de escala 1.5.

Imagem final:	0	1	2	3	4	5
Imagem inicial:	0	0.67	1.33	2	2.67	3.33



Redução, NN

- O que acontece quando usamos a estratégia NN para redução?

Redução, NN

- Repare a imagem abaixo:
  - Largura e altura reduzidas pela metade.
  - Imagem novamente ampliada, para visualização, mantendo um fator de escala inteiro.
    - Faremos isso para todas as imagens reduzidas.

Observe bem as bordas...

Redução, NN

- A redução com vizinhos mais próximos sub-amostra a imagem.
  - Objetos e bordas com poucos pixels de largura podem desaparecer.

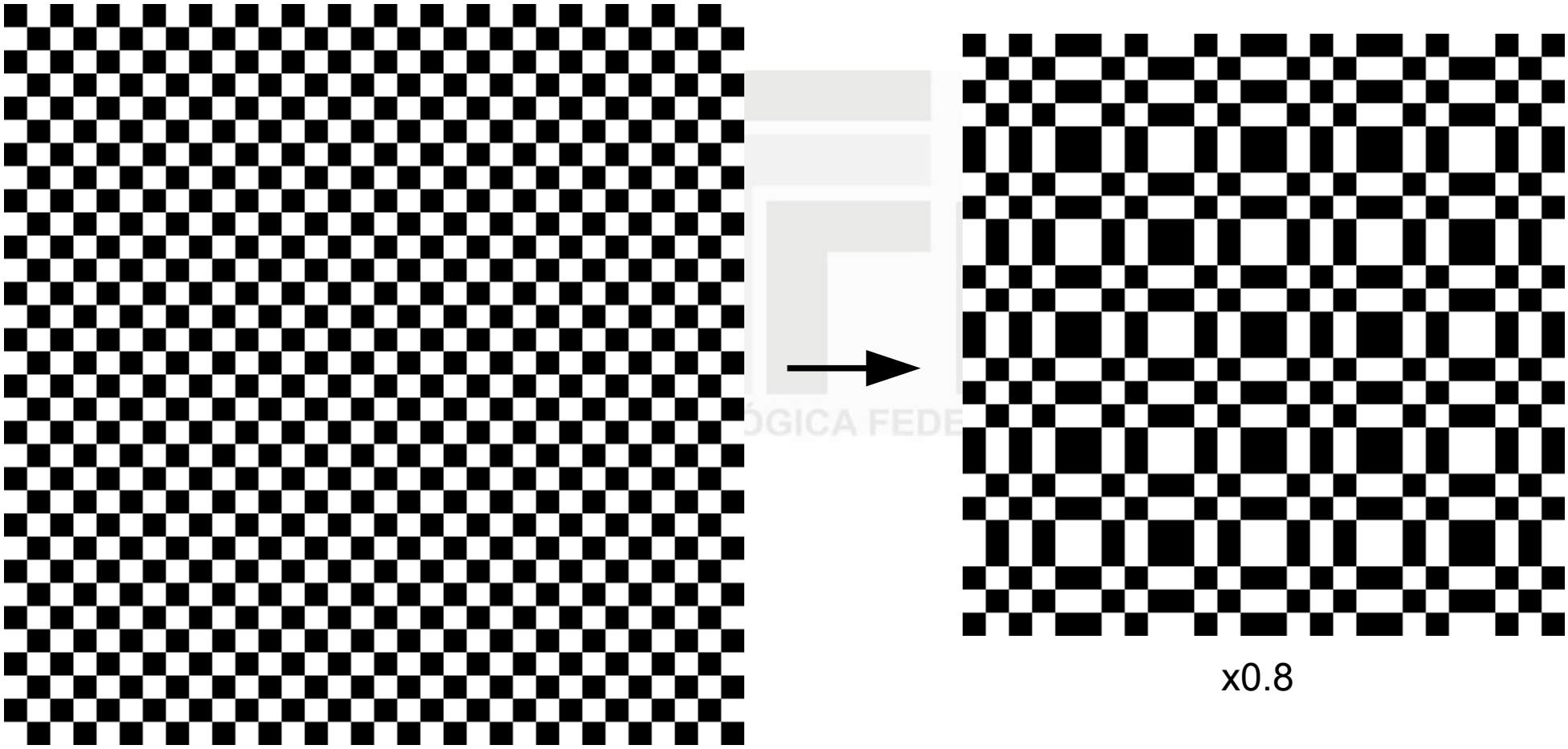
Original

x0.8

x0.5

Redução, NN

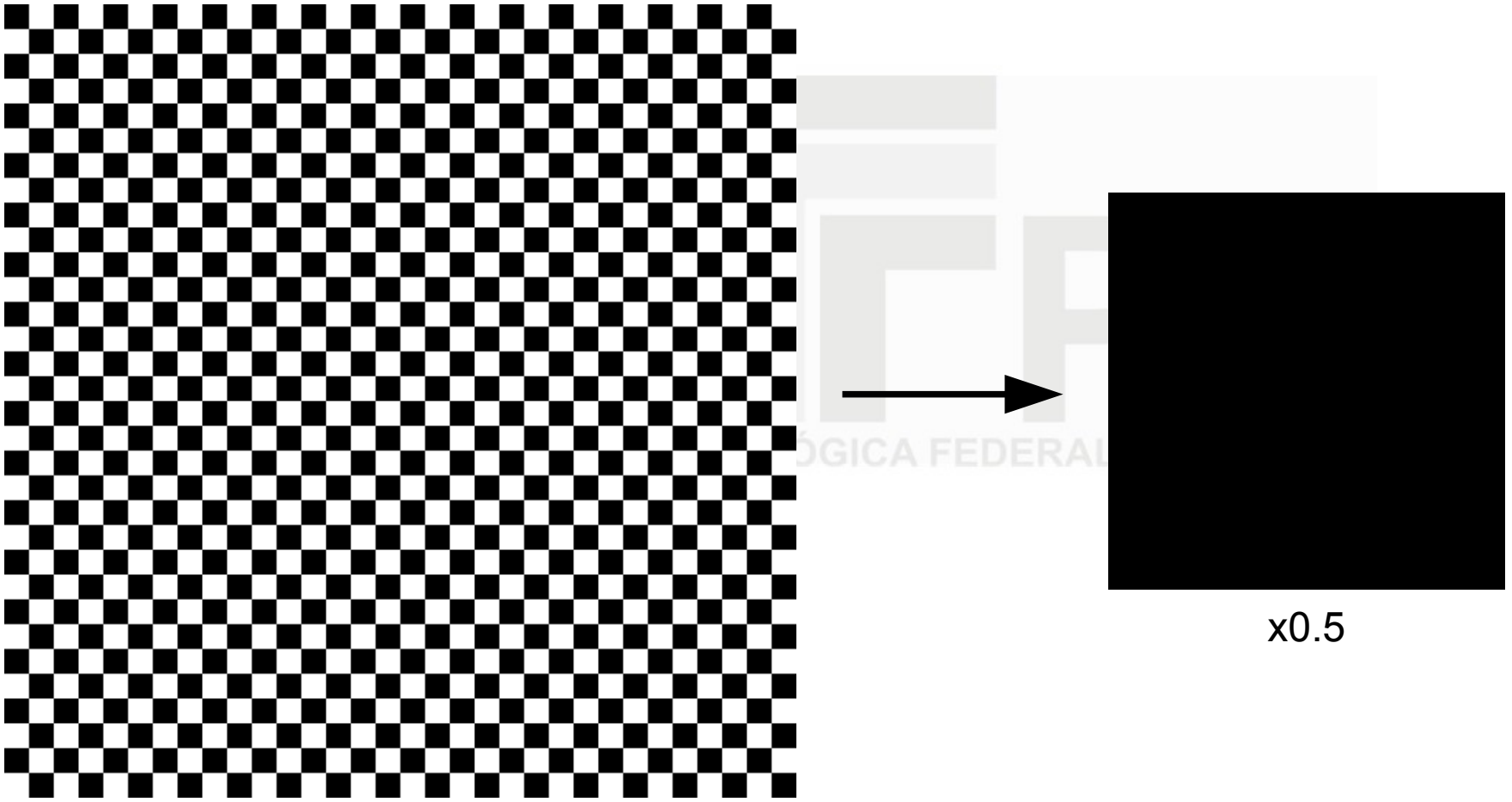
- Cada “quadradinho” abaixo tem 1 pixel de largura/altura.
  - O que acontecerá se o fator de escala for 0.5?



x0.8

Redução, NN

- Esta é uma instância do fenômeno chamado *aliasing*.
  - = um sinal “se disfarça” e fica indistinguível de outro sinal.

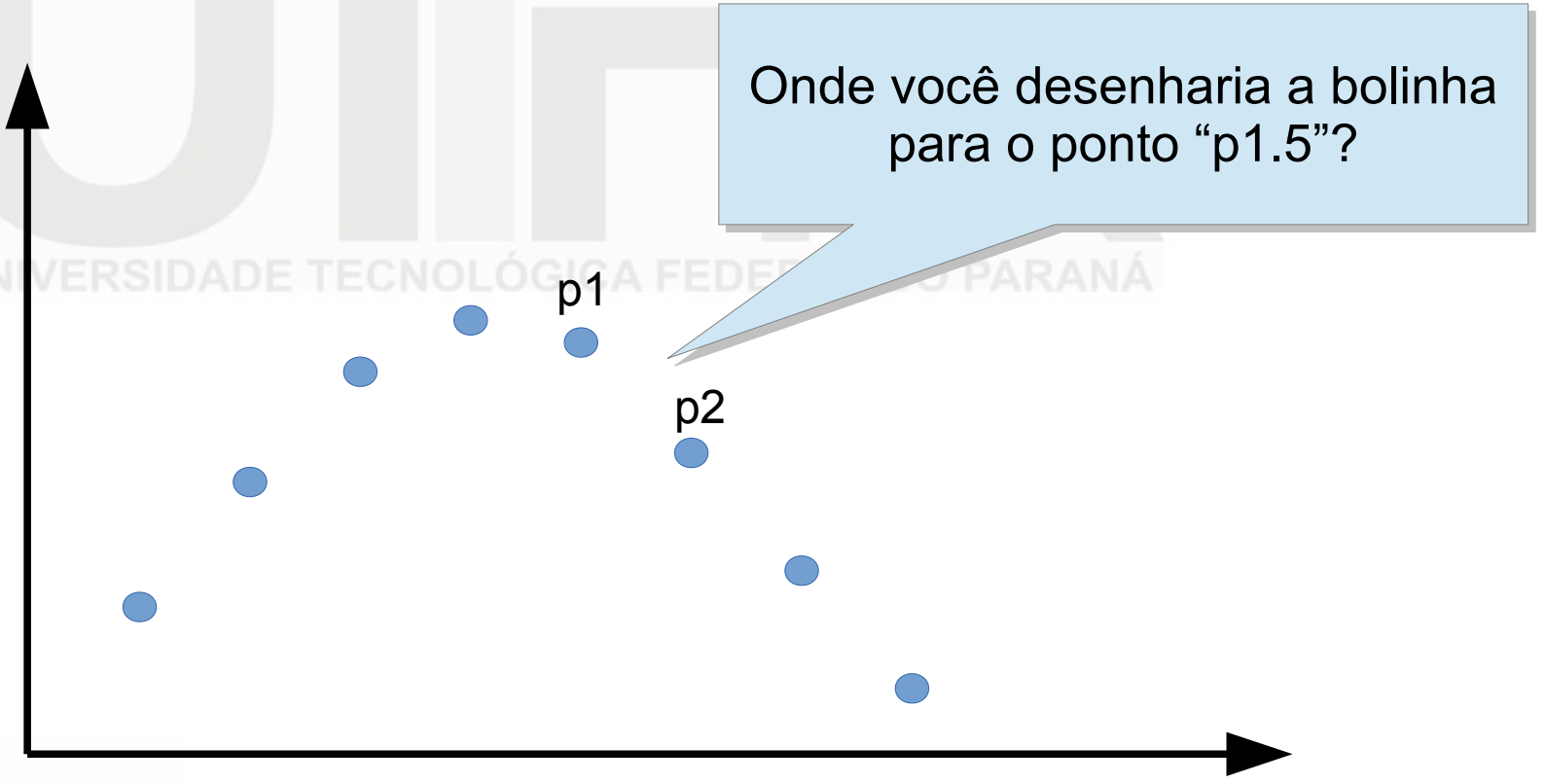


Interpolação

- O que é interpolar?

Interpolação

- O que é interpolar?
  - Em português: intercalar, inserir entre.
- *Interpolação:* determinar o valor de uma função em um ponto dados os valores dos pontos ao seu redor.



Interpolação

- Suponha que você tem um vetor de amostras, mas só conhece o valor das amostras pares.
  - Como você “chutaria” os valores das amostras ímpares?

0	1	2	3	4	5	6	7	8	9
0.1		0.3		0.4		0.4		0.2	

Interpolação

- Agora imagine que queremos ampliar uma imagem, dobrando a sua altura e largura.
  - A imagem é uma função 2D amostrada!
- Como usar o conceito de interpolação para “adivinhar” o valor dos novos pixels?

0	1	0	1	0
0	0.8	0	0.8	0
0	0.2	0.4	0.6	0.8
0	0	0	0	0
1	1	1	1	1

0		1		0		1		0	
0		0.8		0		0.8		0	
0		0.2		0.4		0.6		0.8	
0		0		0		0		0	
1		1		1		1		1	

Interpolação

- Cada pixel “novo” recebe a média de até 4 valores.
- Como poderíamos usar este mesmo princípio para ampliar uma imagem com um fator de escala arbitrário?

Interpolação bilinear

- Usamos a mesma ideia do redimensionamento com vizinhos mais próximos, mas sem truncar as coordenadas.
  - $g(x,y) = f(x/s, y/s)$.
- Exemplo:
  - Para um fator de escala 2, o pixel (100, 80) da imagem de saída será igual ao pixel (50, 40) da imagem de entrada.
  - Para um fator de escala 1.4. o pixel (100, 80) da imagem de saída será igual ao pixel... (?)

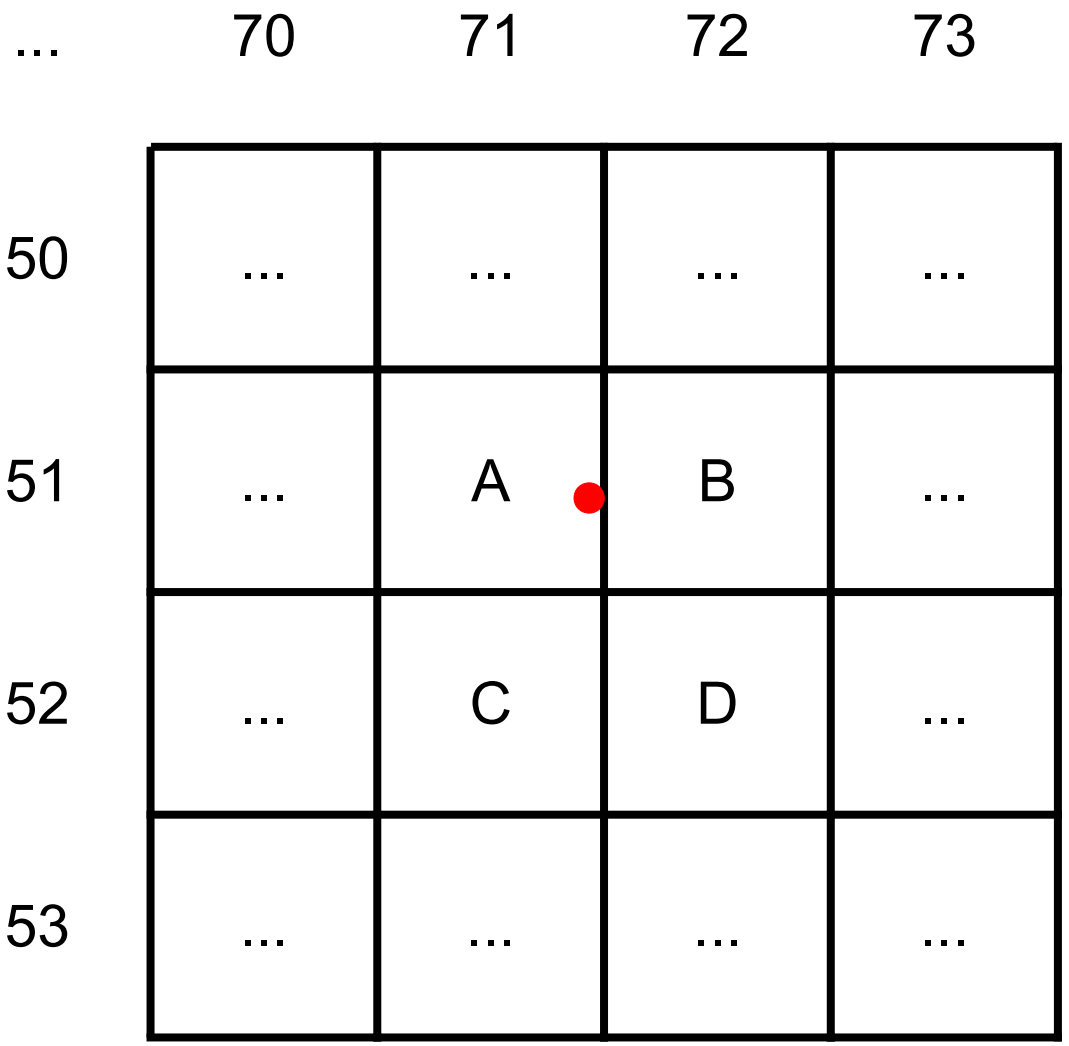
Interpolação bilinear

- Usamos a mesma ideia do redimensionamento com vizinhos mais próximos, mas sem truncar as coordenadas.
  - $g(x,y) = f(x/s, y/s)$.
- Exemplo:
  - Para um fator de escala 2, o pixel (100, 80) da imagem de saída será igual ao pixel (50, 40) da imagem de entrada.
  - Para um fator de escala 1.4. o pixel (100, 80) da imagem de saída será igual ao pixel (71.43, 51.14) da imagem de entrada.
    - Mas este pixel não existe!!!
    - A abordagem NN truncaria para (71, 51).
    - Como “adivinhar” o valor do pixel (71.43, 51.14)?

Exemplo anterior

Supondo que as coordenadas indicam o centro de cada pixel, a bolinha vermelha mostra onde estaria o pixel (71.43, 51.14).

Qual seria o valor do pixel nesta posição?



- Na interpolação bilinear, descobrimos o valor de um pixel fazendo a média ponderada dos 4 pixels mais próximos.
  - $g(x,y) = f(x/s, y/s) = f(x', y') = w_1 \cdot f(\text{floor}(x'), \text{floor}(y')) +$
    $w_2 \cdot f(\text{floor}(x'), \text{ceil}(y')) +$
    $w_3 \cdot f(\text{ceil}(x'), \text{floor}(y')) +$
    $w_4 \cdot f(\text{ceil}(x'), \text{ceil}(y'))$

- Para obter os pesos w_n:
  - w_l (à esquerda): ceil $(x') - x'$
  - w_r (à direita): $x' - \text{floor}(x')$
  - w_t (acima): ceil $(y') - y'$
  - w_b (abaixo): $y' - \text{floor}(y')$
  - $w_1 = w_l \cdot w_t$
  - $w_2 = w_l \cdot w_b$
  - $w_3 = w_r \cdot w_t$
  - $w_4 = w_r \cdot w_b$

Note que:

$w_l + w_r = 1$

$w_t + w_b = 1$

Então, $w_1 + w_2 + w_3 + w_4 = 1$

Aplicando para o exemplo

$g\,(100, 80) = f\,(71.43, 51.14)$

$f\,(71, 51) = A$
$f\,(72, 51) = B$
$f\,(71, 52) = C$
$f\,(72, 52) = D$

$w_l = 72 - 71.43 = 0.57$

$w_r = 71.43 - 71 = 0.43$

$w_t = 52 - 51.14 = 0.86$

$w_b = 51.14 - 51 = 0.14$

$$g\,(100, 80) = A \cdot w_l \cdot w_t + B \cdot w_r \cdot w_t + C \cdot w_l \cdot w_b + D \cdot w_r \cdot w_b$$

Interpolação bilinear: ideia

```
dados os fatores de escala horizontal e vertical sh e sv

para cada linha y da imagem de saída
    y' = y/sv
    wb = y' - floor (y')
    wt = 1 - wb

    para cada coluna x da imagem de saída
        x' = x/sh
        wr = x' - floor (x')
        wl = 1 - wr

        g (x,y) = f (floor (x), floor (y)) * wl * wt +
                  f (ceil (x), floor (y)) * wr * wt +
                  f (floor (x), ceil (y)) * wl * wb +
                  f (ceil (x), ceil (y)) * wr * wb
```

É importante aqui ir da saída para a entrada, e não o contrário!

Margens

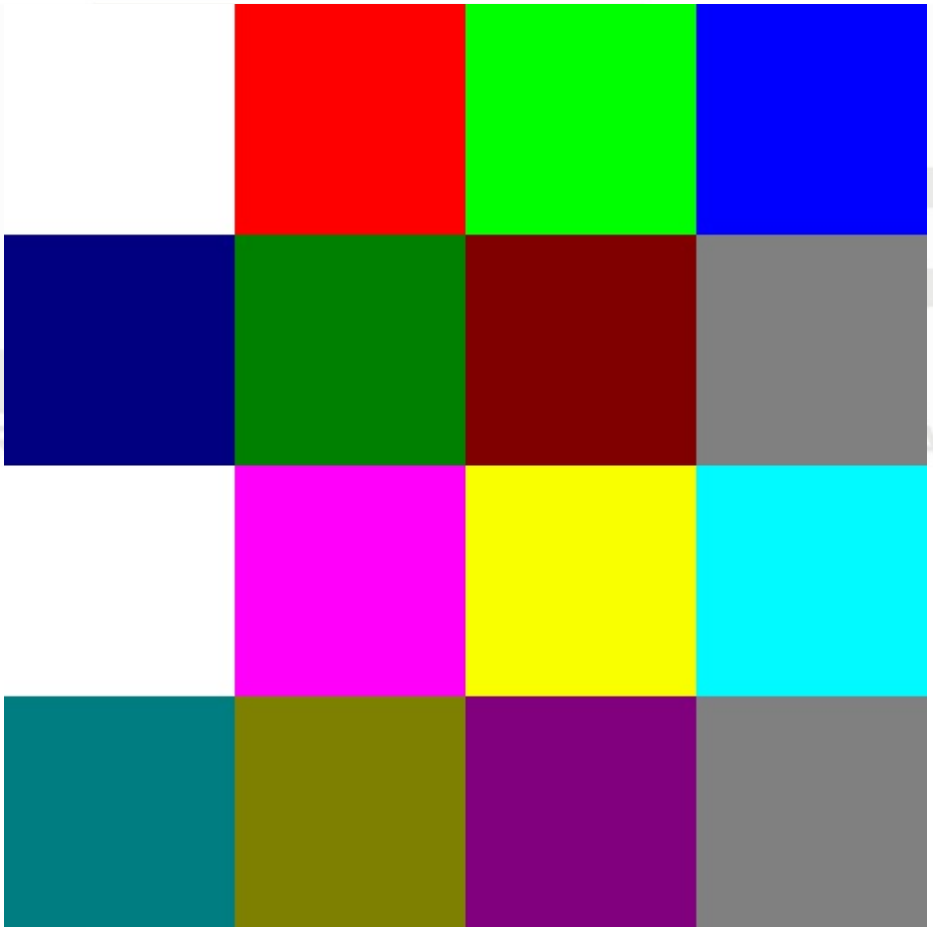
- Suponha que a entrada tem 100 pixels de largura, e um fator de escala s=1.5.
- A imagem de saída terá 150 pixels de largura.
- Os pixels na última coluna da imagem de saída (x = 149) terão os pesos calculados a partir do pixel x/s = 99.333...
- Existe um problema aqui... qual?

Margens

- Suponha que a entrada tem 100 pixels de largura, e um fator de escala s=1.5.
- A imagem de saída terá 150 pixels de largura.
- Os pixels na última coluna da imagem de saída (x = 149) terão os pesos calculados a partir do pixel x/s = 99.33...
- Existe um problema aqui... qual?
  - A última coluna vai ter valores calculados com base em ceil (99.33) = 100 – uma posição que não existe na imagem de entrada!
- Para tratar disso, podemos considerar:
  - Uma margem preta ao redor da imagem.
  - Que a imagem se repete em um padrão estilo “azulejos” (*tiles*).
  - Que a imagem se repete de forma espelhada.

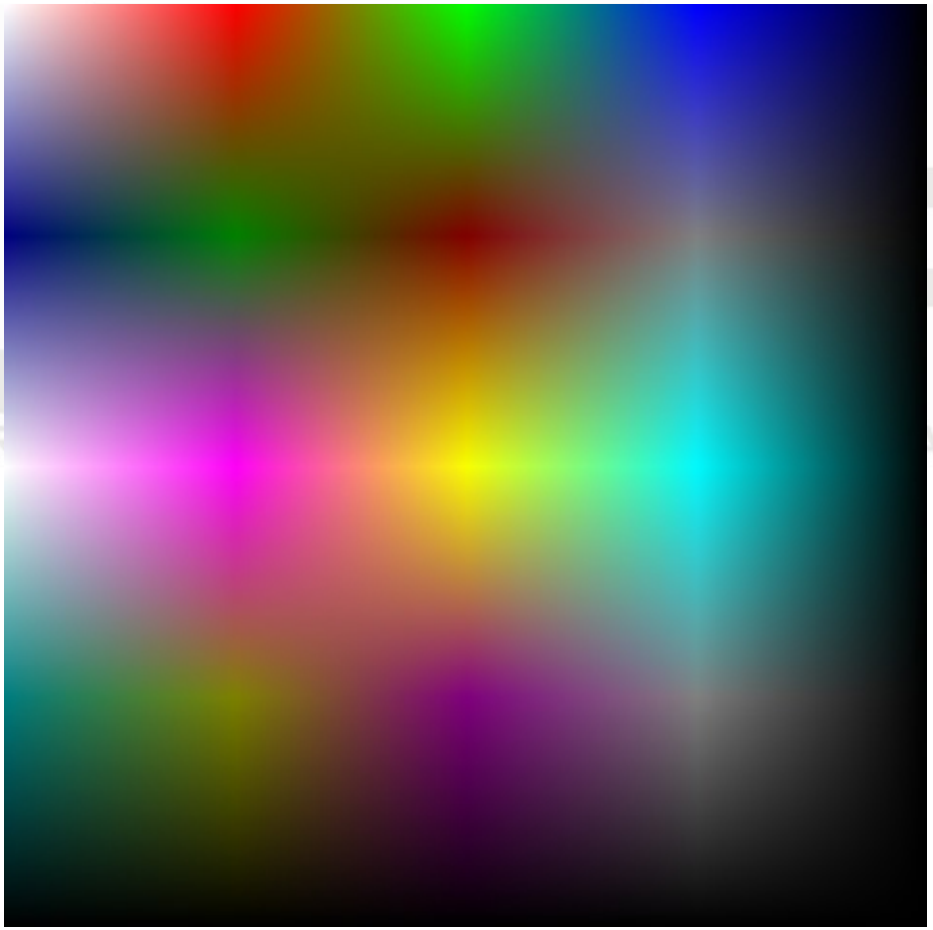
Exemplo

- Considere um padrão com 4x4 pixels, ampliado 120 vezes.
  - (primeiro com NN).



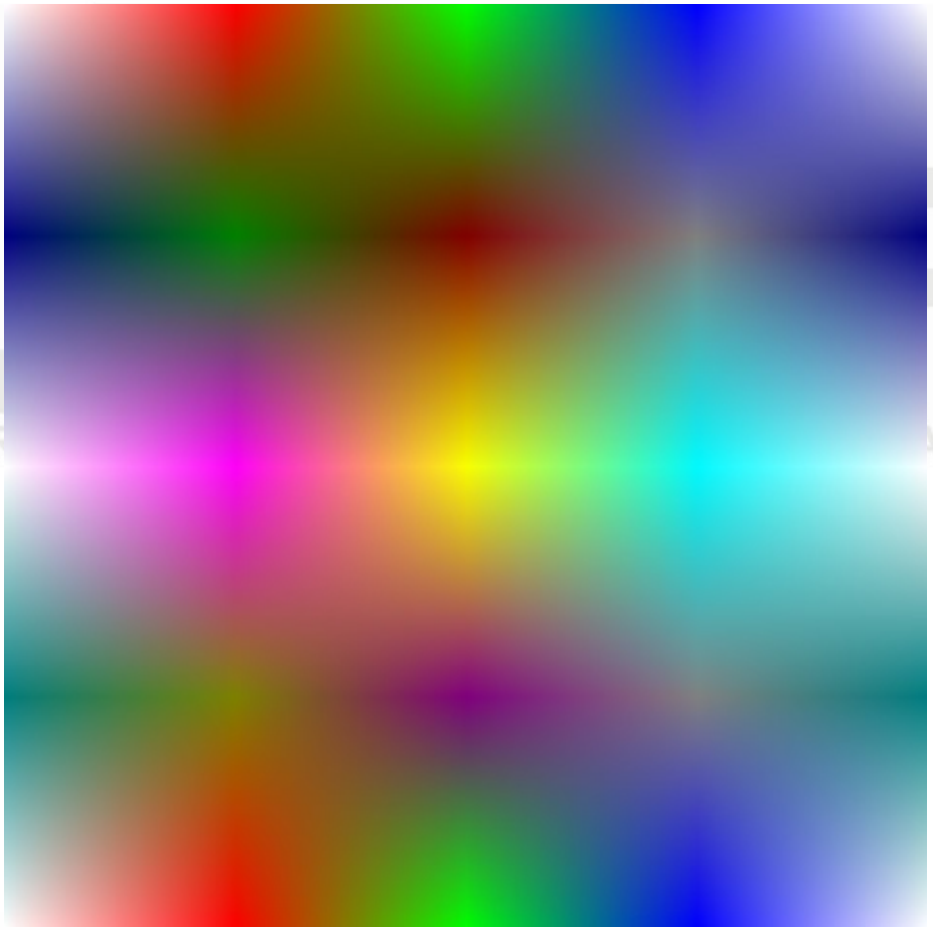
Exemplo

- Supondo uma margem preta...



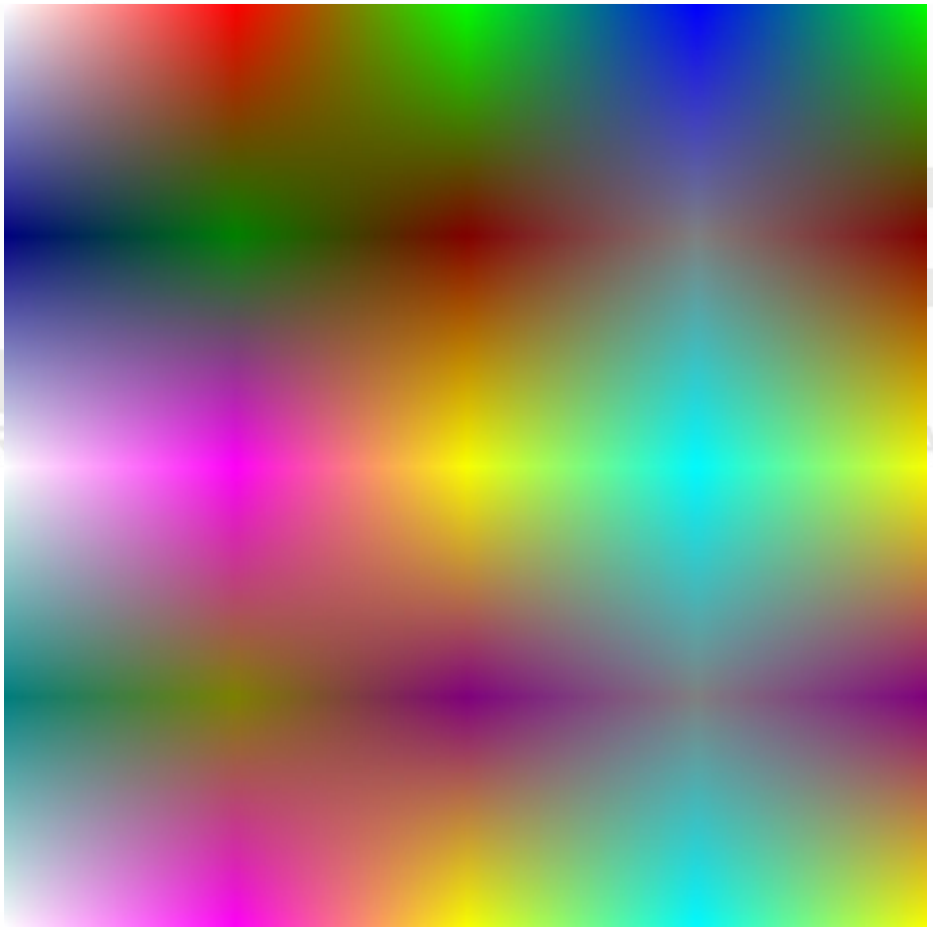
Exemplo

- Com repetições infinitas (*tiled*).



Exemplo

- Com repetições espelhadas.



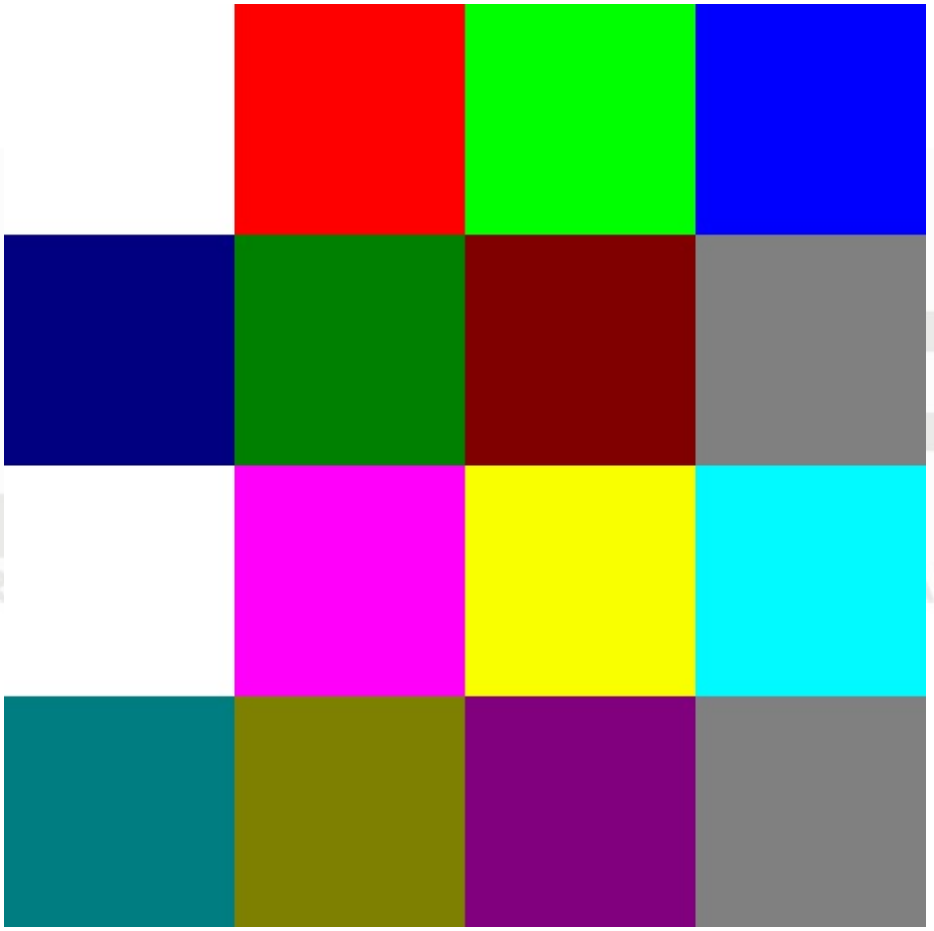
Margens... e outro problema

- Supomos daqui para a frente o esquema com repetições infinitas.
- Mas surgiu outro problema.
  - Você notou algo de estranho com o alinhamento dos pixels com cores “puras”?

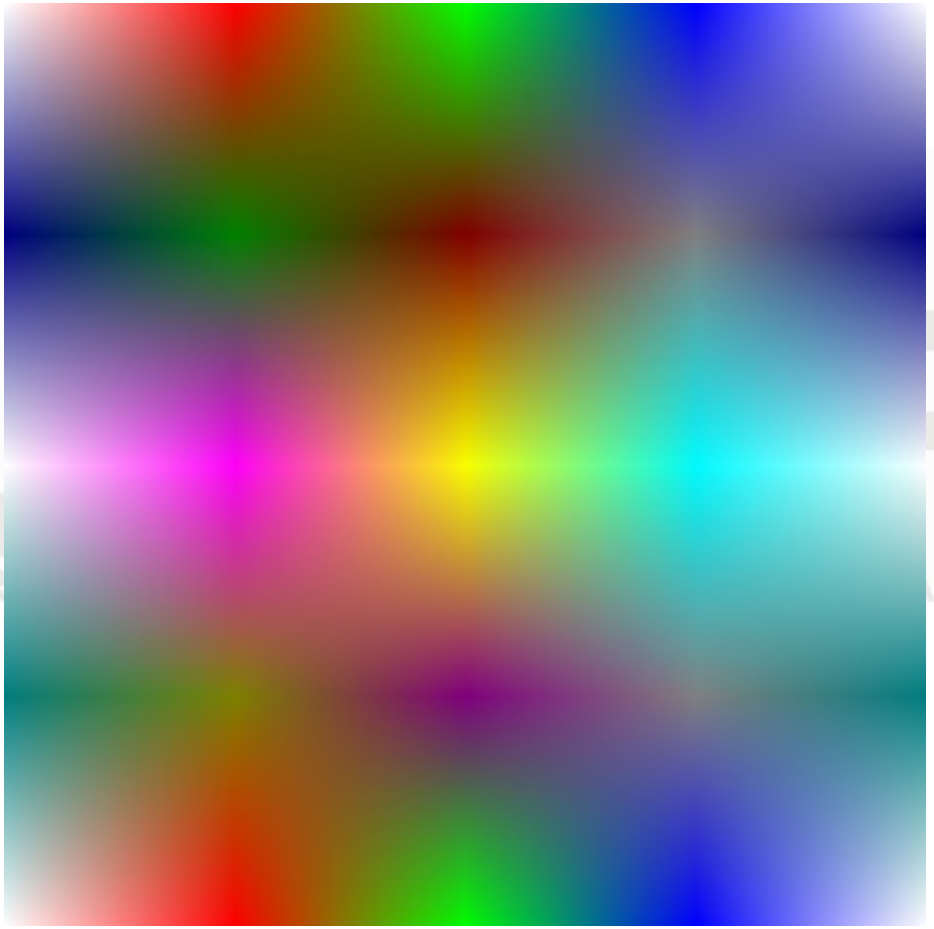
Margens... e outro problema

- Supomos daqui para a frente o esquema com repetições infinitas.
- Mas surgiu outro problema.
  - O pixel f(0,0) foi mapeado para g(0,0).
  - O pixel f(0,1) foi mapeado para g(0,120).
  - O pixel f(0,2) foi mapeado para g (0,240).
  - O pixel f(0,3) foi mapeado para g (0,360).
- Se imaginarmos um grid sobre a imagem de saída, os pixels originais foram mapeados para os cantos do grid!
- Para mapear os pixels originais para os centros do grid, basta subtrair 0.5 das coordenadas na imagem de entrada:
  - $g(x, y) = f(x/s - 0.5, y/s - 0.5)$

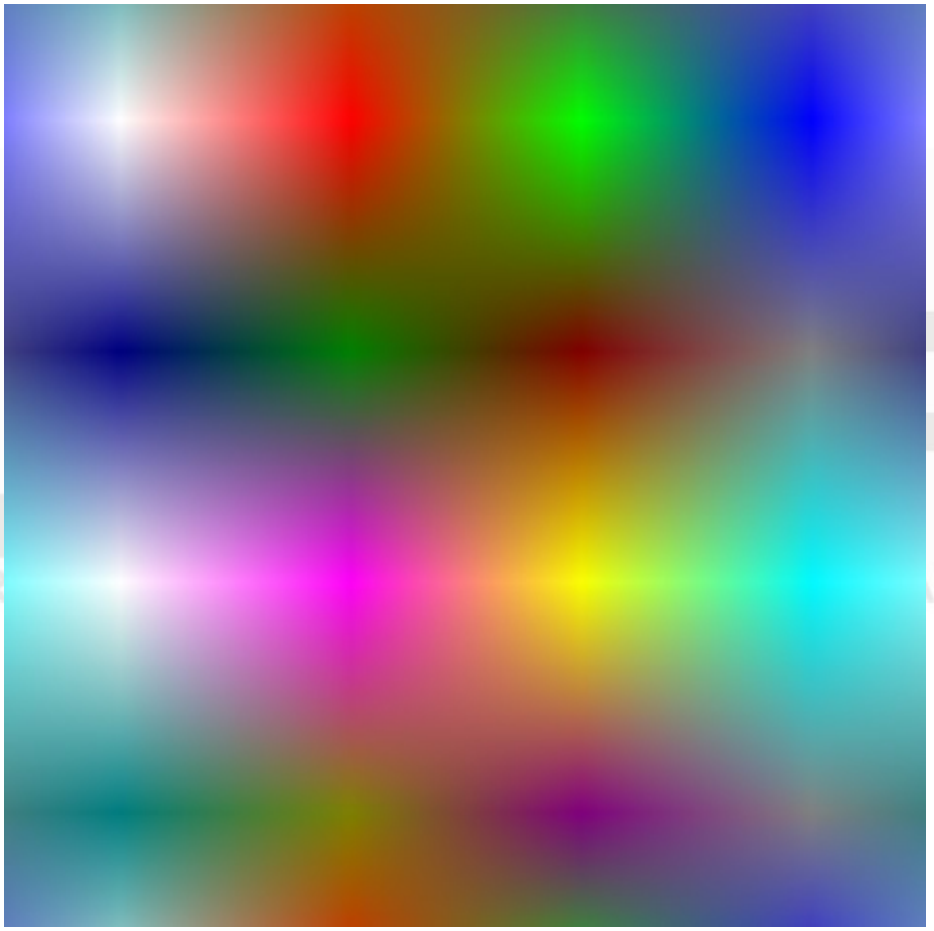
Com NN



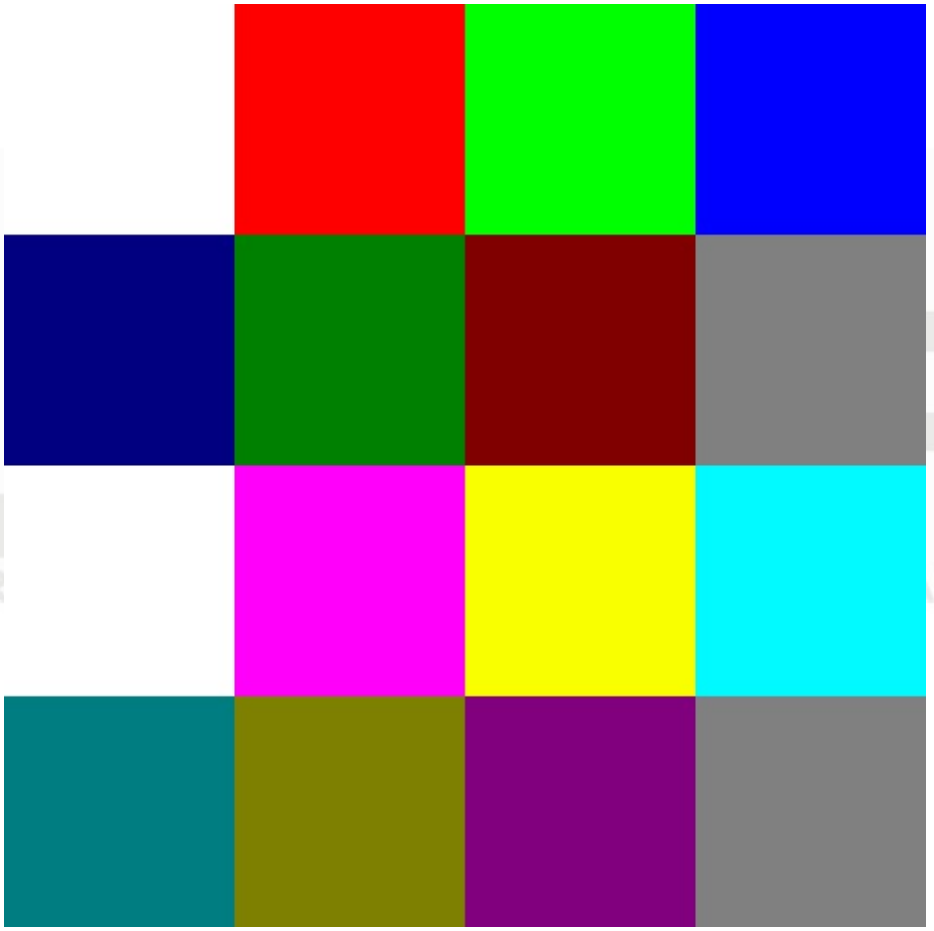
Interpolação bilinear (sem ajuste)



Interpolação bilinear (com ajuste)



Com NN



Redução

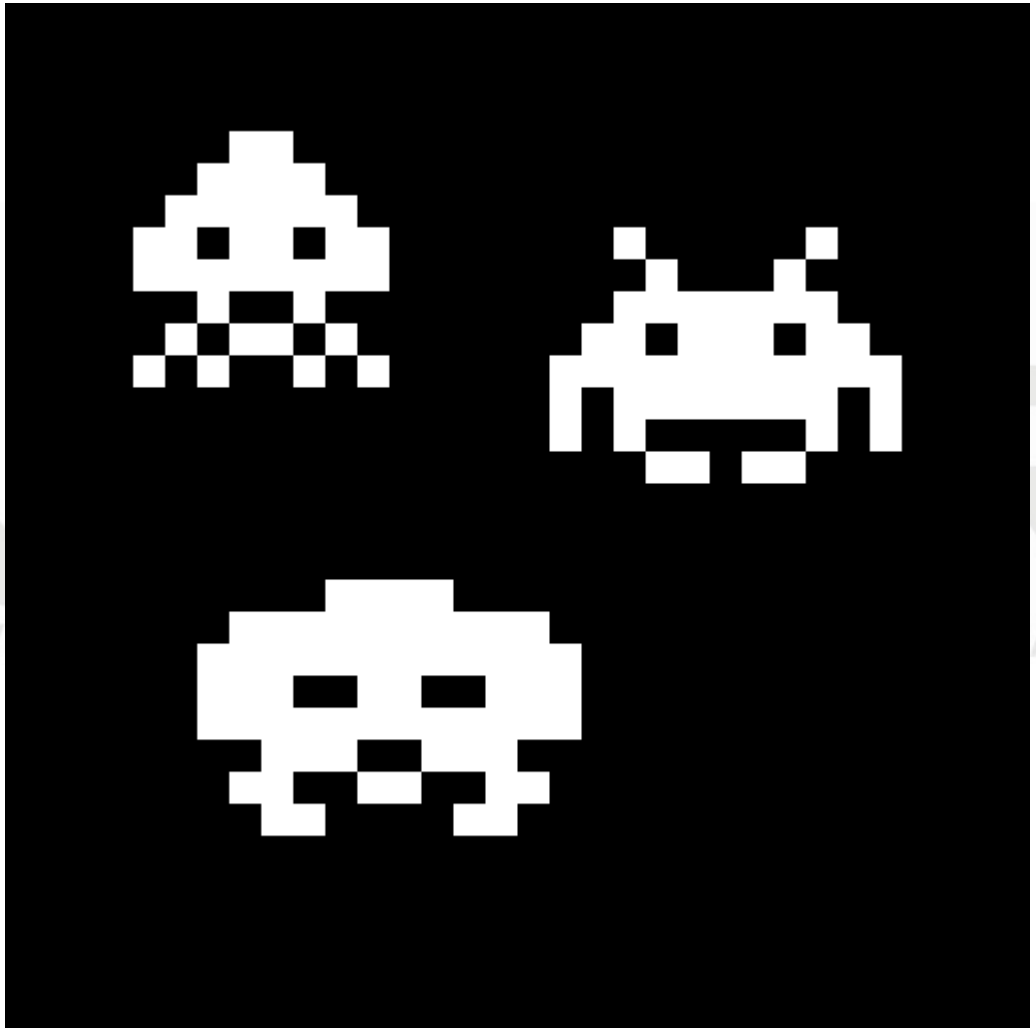
- A interpolação bilinear pode ser usada para reduzir imagens.
- Ainda podemos ter pixels completamente descartados, mas os pixels restantes podem vir de mais de um pixel original.

Original

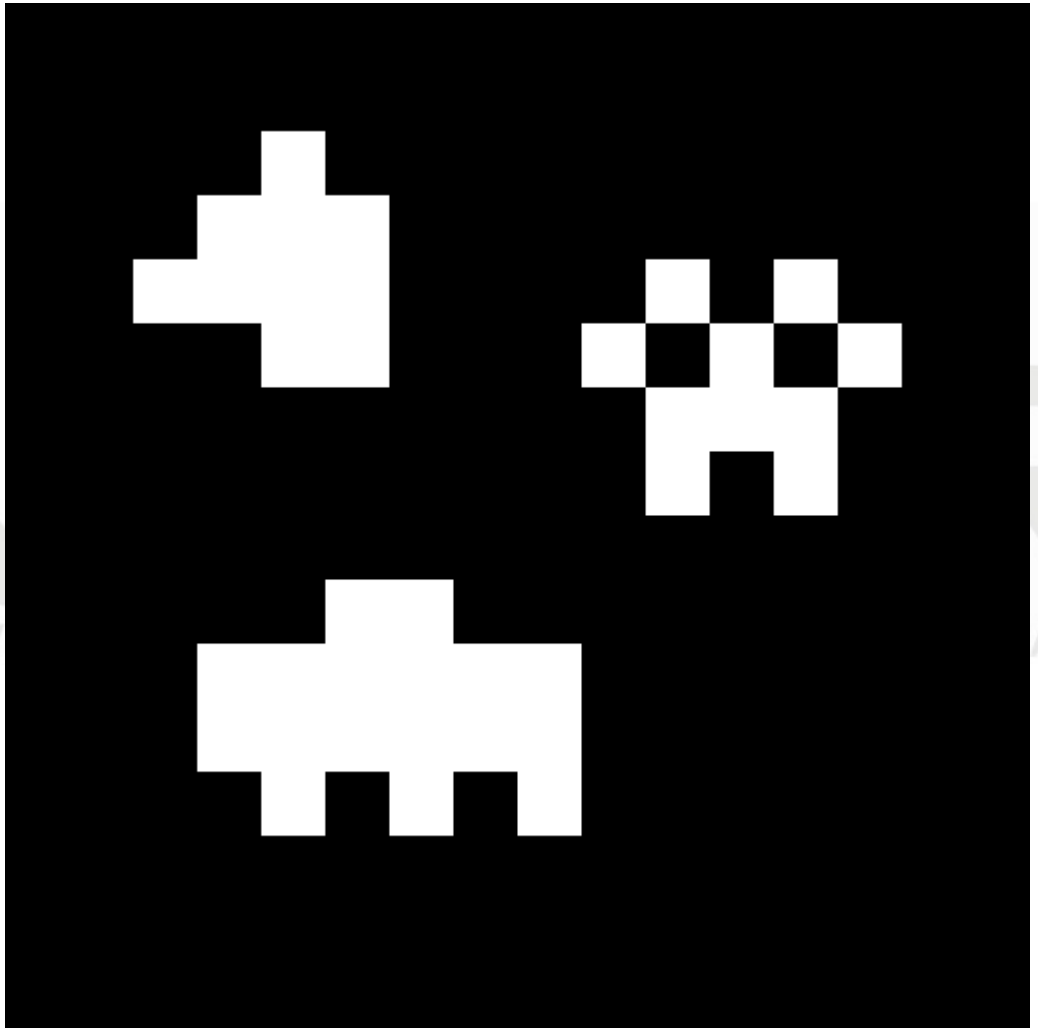
x0.8 (NN)

x0.8 (bilinear)

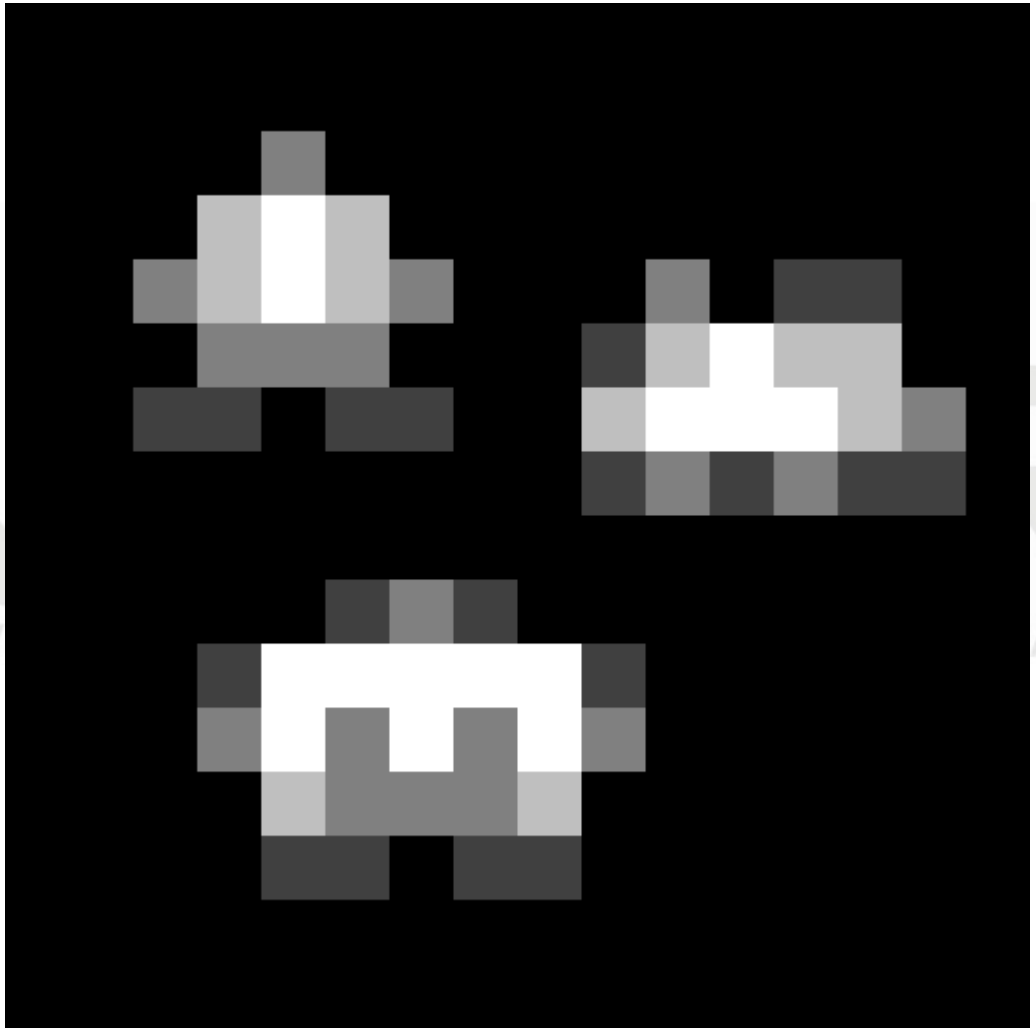
Imagem original (32x32)



Redução com *s*=0.5 (NN)



Redução com *s*=0.5 (bilinear)



Outros métodos...

- Interpolação bicúbica:
  - Usa 16 pixels em uma vizinhança 4x4.
  - Ajustando uma superfície sobre os pontos conhecidos na entrada.
  - Preserva detalhes melhor que a interpolação bilinear, mas tem custo computacional mais alto, e pode gerar artefatos.
- Lanczos:
  - Método baseado na convolução com um kernel.
  - Mais sofisticado, é usado por alguns algoritmos de compressão.
  - Pode gerar artefatos.
- Métodos especializados para *pixel art*.
  - Usados em emuladores de video games.
- Métodos baseados em borrar-e-subamostrar.
  - Usados para reduzir imagens usando dados de todos os pixels.

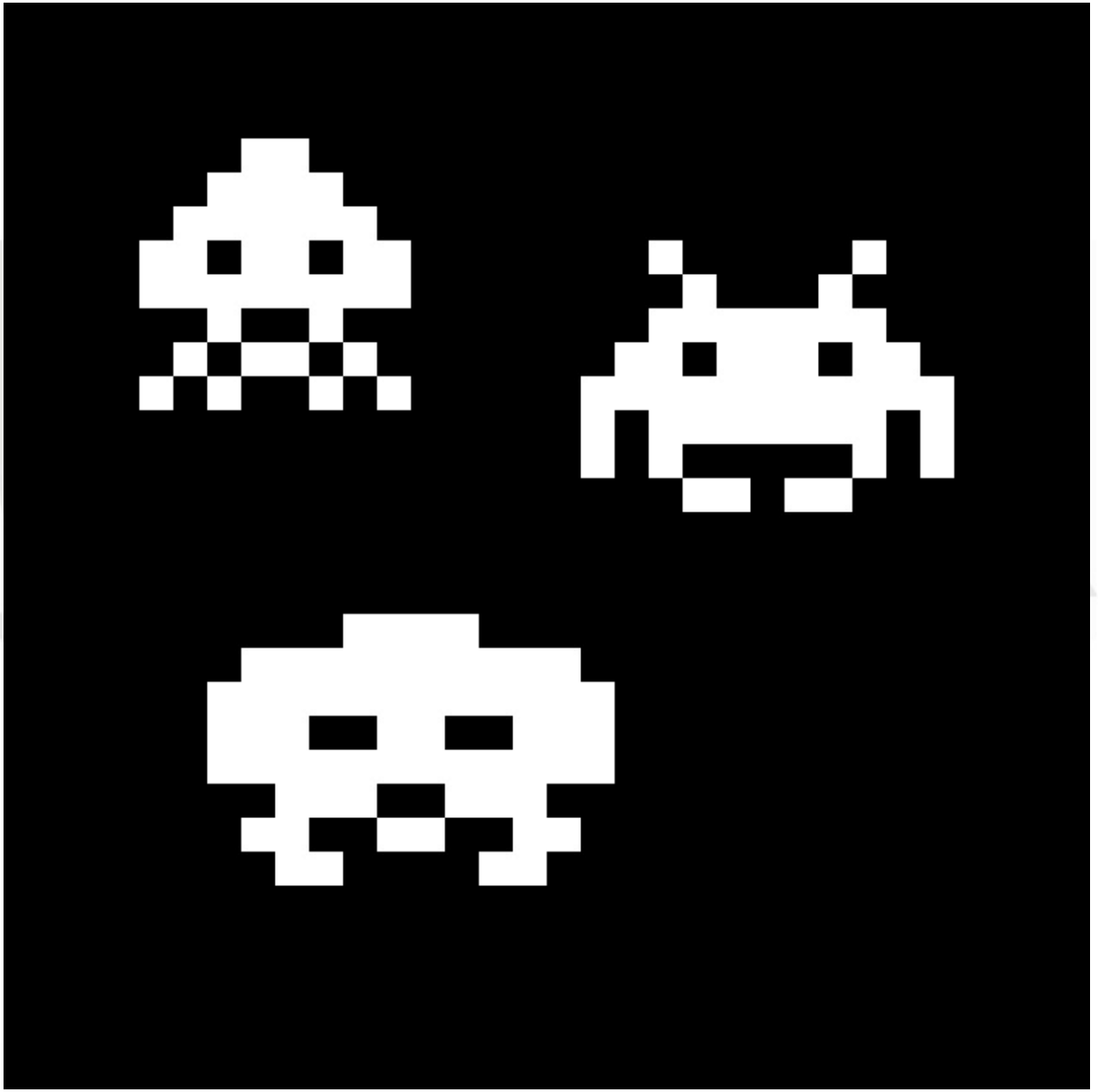
Tamanho x3, NN

Tamanho x3, bilinear

Tamanho x3, bicúbica

Tamanho x3, Lanczos

Tamanho x20, NN



Tamanho x20, bilinear

Tamanho x20, bicúbica

Tamanho x20, Lanczos

Tamanho x2.5, NN

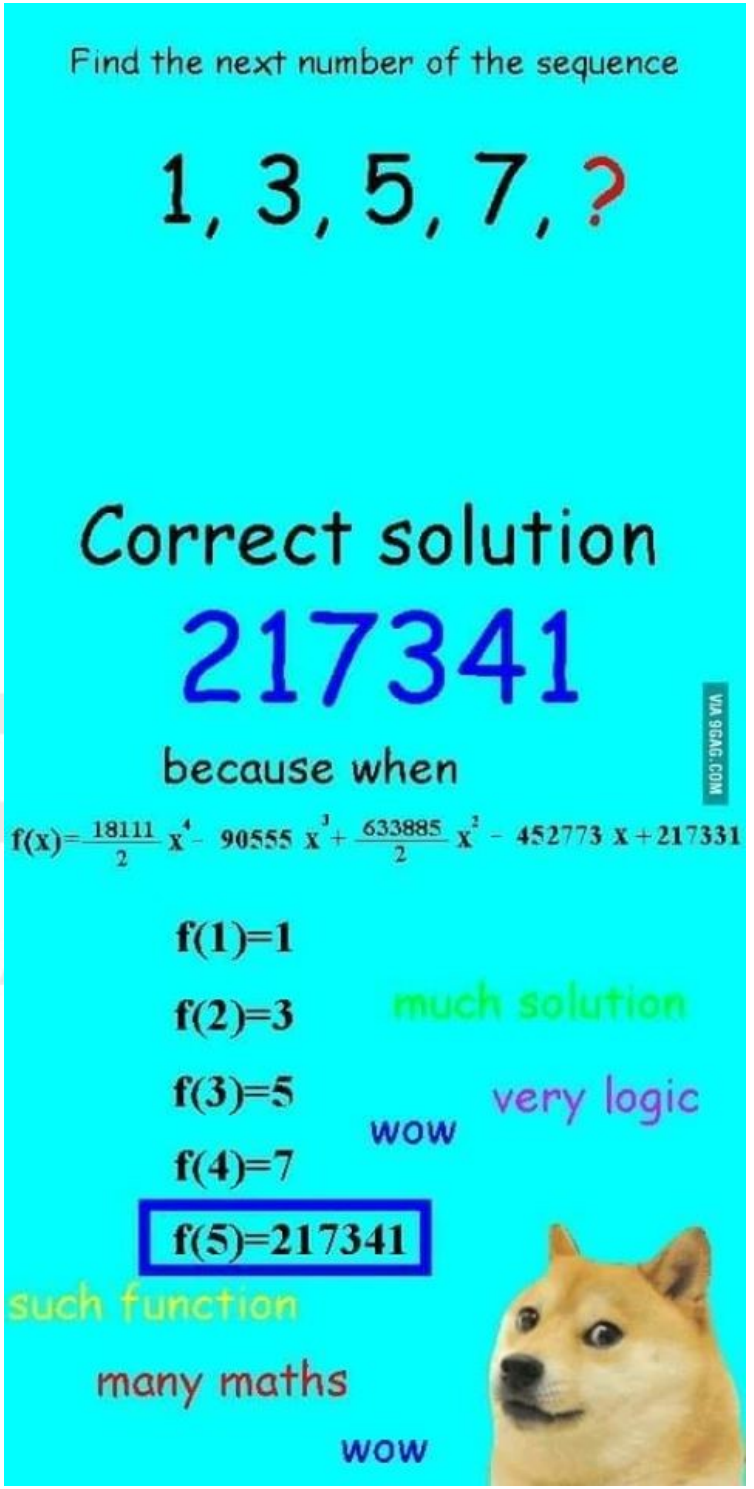
Tamanho x2.5, bilinear

Tamanho x2.5, bicúbica

Tamanho x2.5, lanczos

“With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.”

- John Von Neumann.



Interpolação: notas finais

- As técnicas vistas podem ser usadas para outras transformações geométricas, além da escala!
- Curiosidade:
  - Nintendo 64: memória de texturas muito pequena, mas o hardware implementa um algoritmo de interpolação linear de 3 pontos.
  - Playstation: muito mais memória disponível para texturas, mas não tem unidade de ponto flutuante, e só usa a abordagem NN.

Playstation

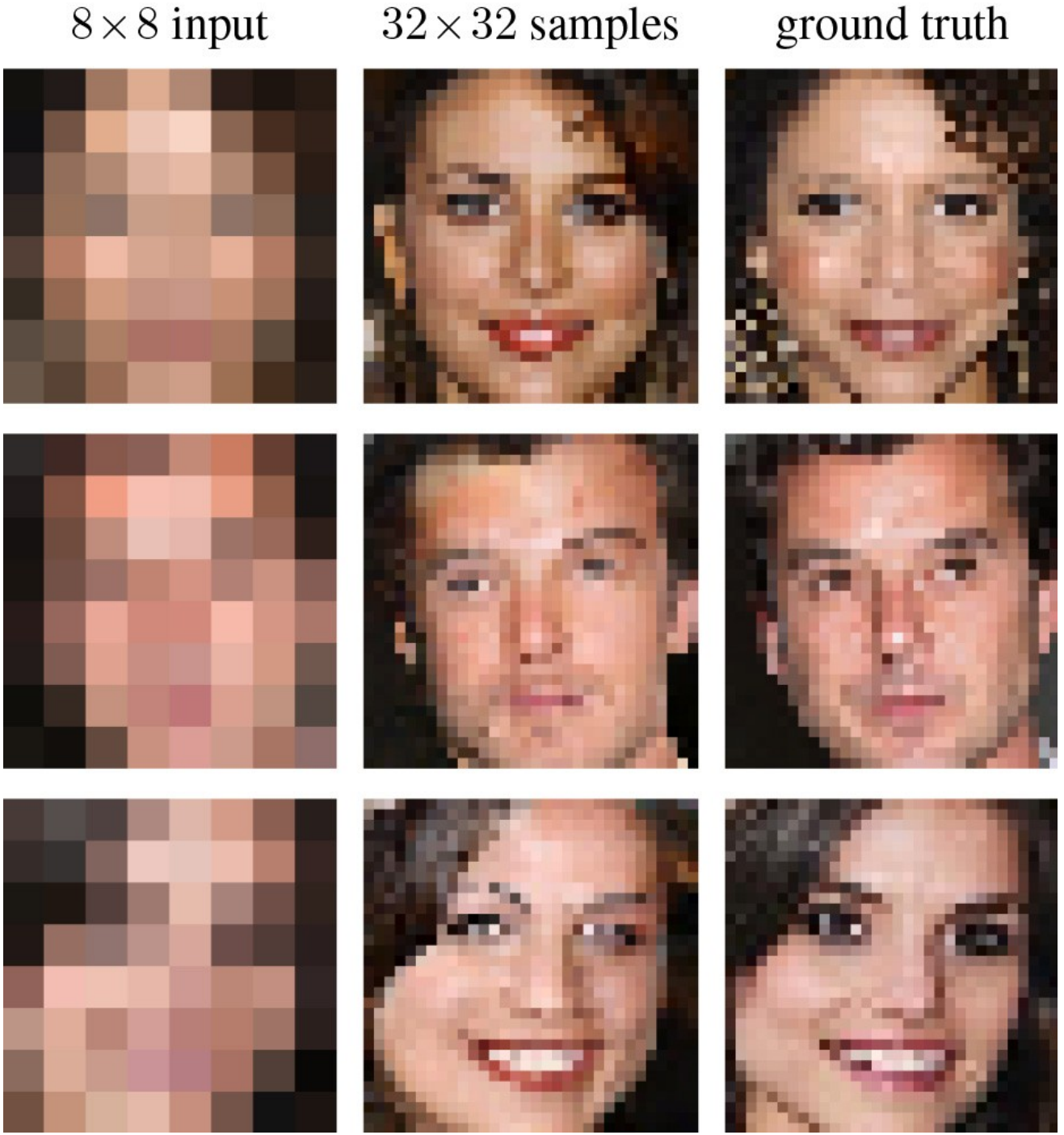
Nintendo 64

Superresolução

- Superresolução: técnicas que procuram preservar / recuperar a qualidade de imagens com resolução ampliada.

- Ideia mais geral:
  - Explorar redundâncias.
  - Obter várias imagens, com pequenas variações de enquadramento.
  - Alinhar as imagens.
    - *Registro* de imagens, tópico que será abordado futuramente.
  - Combinar as imagens em uma imagem de resolução mais alta.
  - Usar técnicas de *deblur* (ou *deconvolução*) para recuperar detalhes.

- Outros tipos de redundância podem ser explorados.
  - Ex: usar aprendizado de máquinas para explorar a similaridade entre imagens diferentes.
  - Ex: reconstrução temporal (PS4 pro, XBox One X).


8 × 8 input
32 × 32 samples
ground truth

a small noisy image

Your image viewer

waifu2x

www.youtube.com/watch?v=OaEgc46FNWE