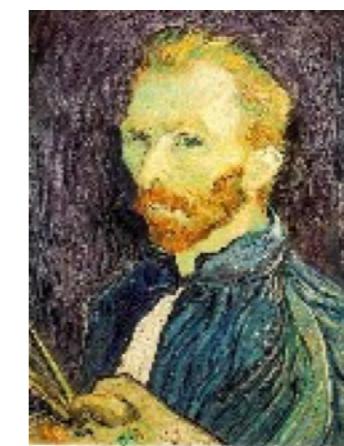
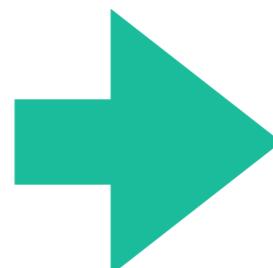
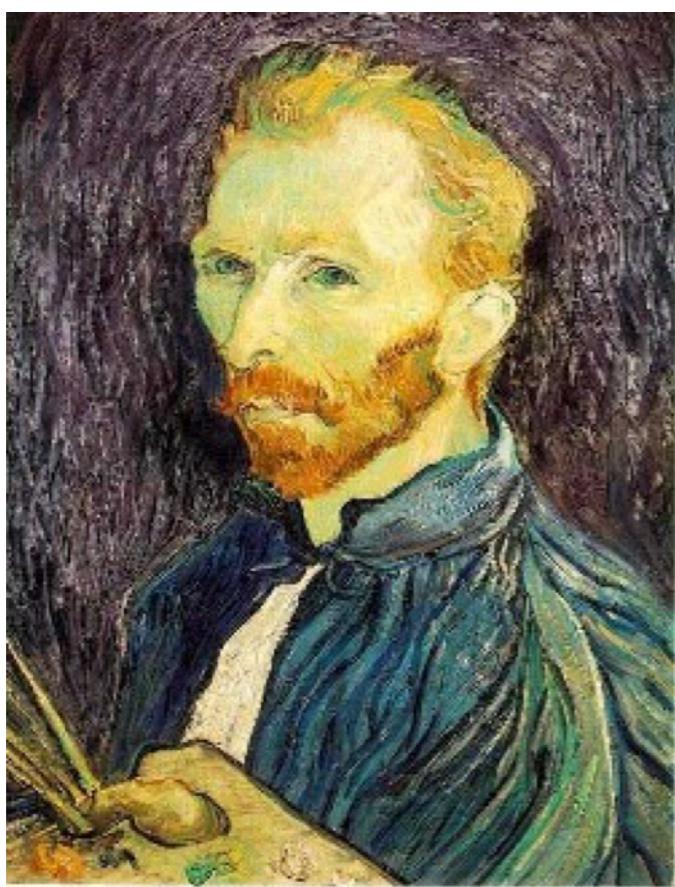
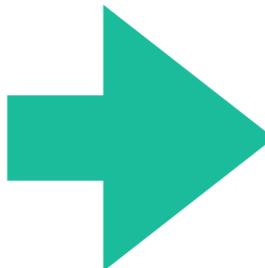


画像処理 リサンプリング

藤田 一寿

■ 画像の拡大縮小をしたい

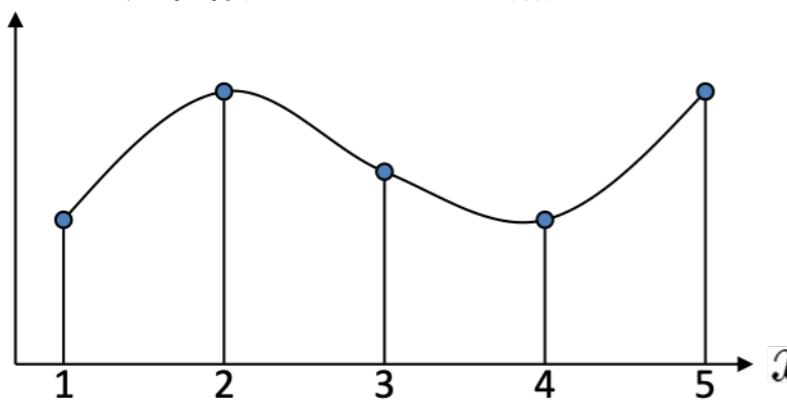


■ 画像拡大とは

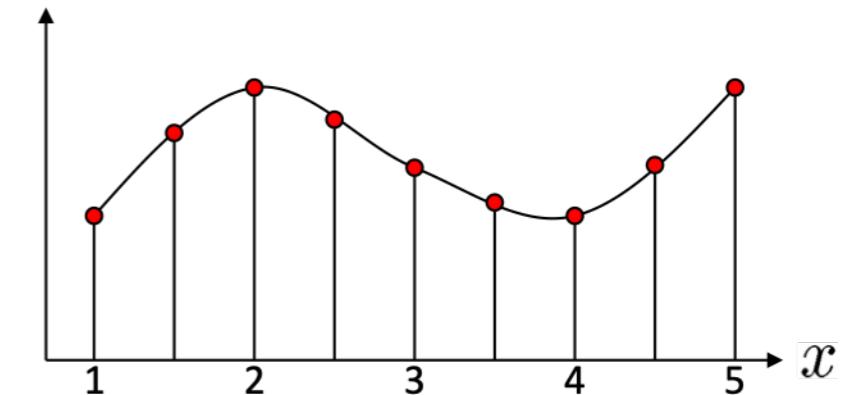
画像拡大とはサンプル数を増やすことに相当する。原画像のサンプル点の間に新しいサンプル点を作ることになる。

無限精度の画像が存在すれば特に問題ないが

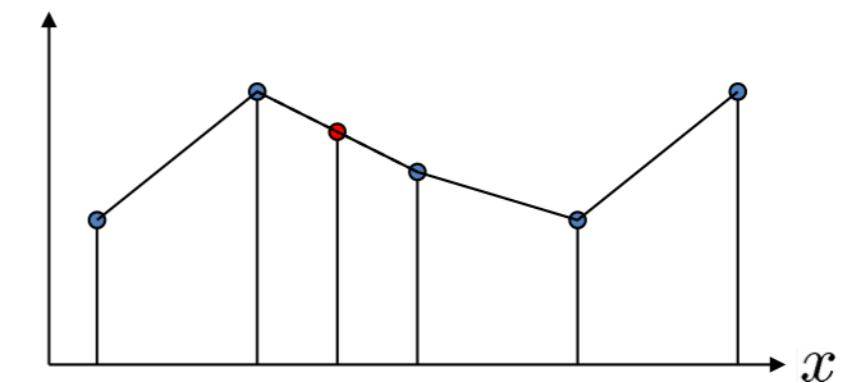
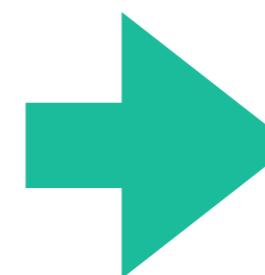
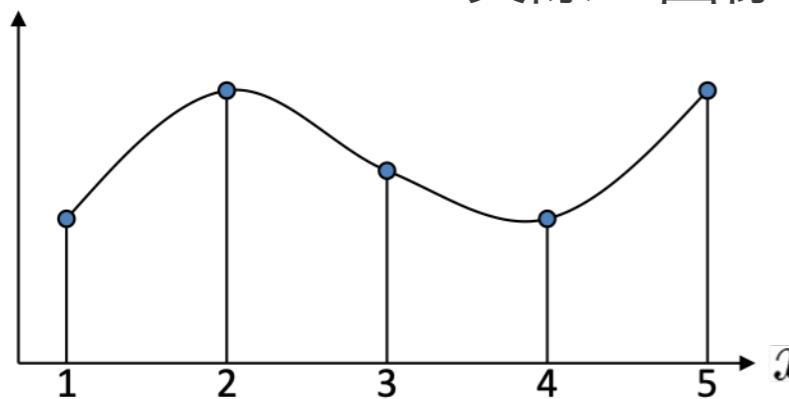
元画像のサンプル点



拡大後の画像のサンプル点



実際の画像ではサンプル点の輝度値しか分からない。



新しい画素の画素値を補完する必要がある！！

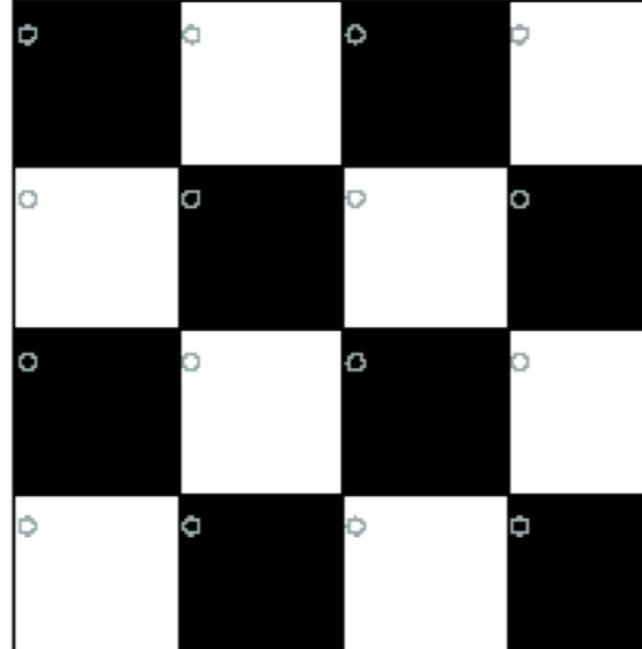
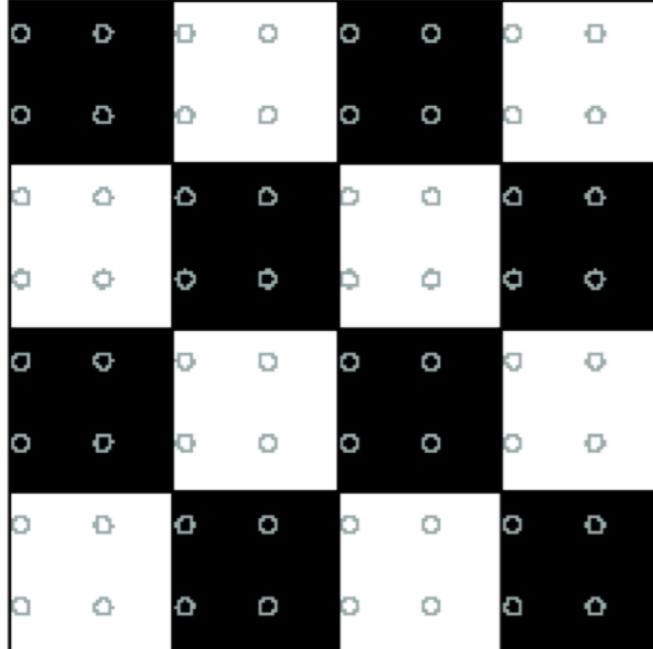
■ 画像縮小の問題



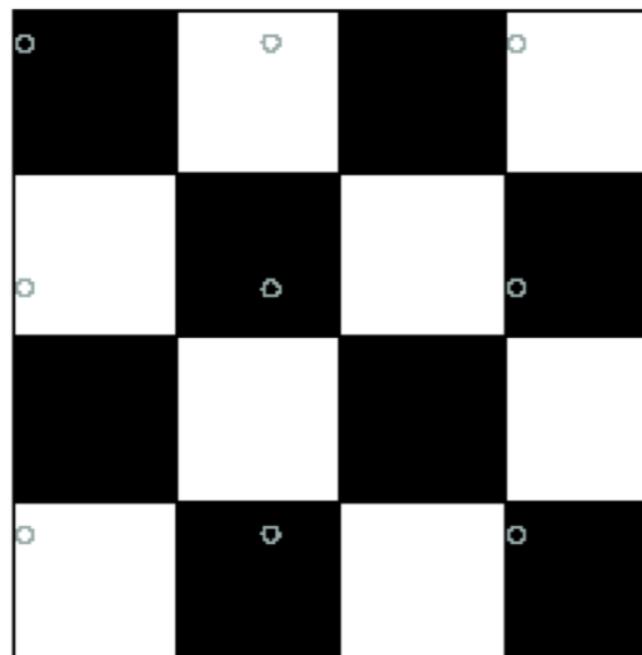
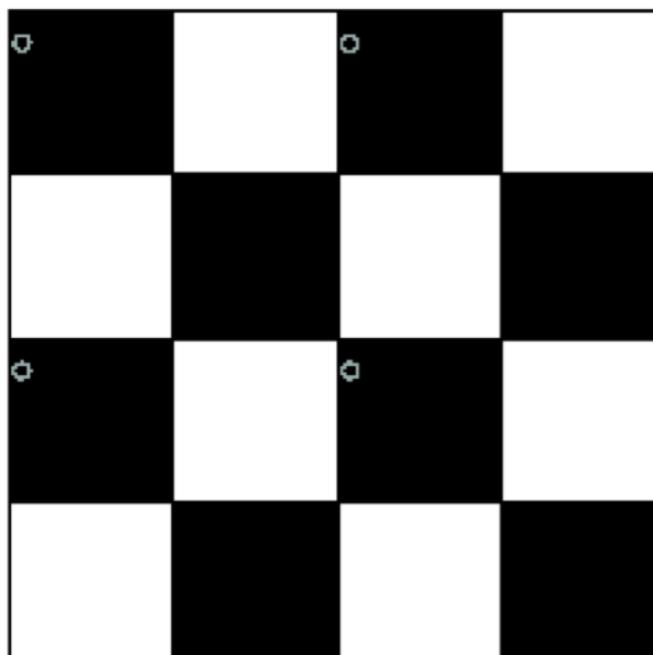
縮小時に変な模様ができるのか

縞模様や格子を下手に縮小すると変な模様が出るのをどう避けるか？

■ 良いサンプリングと悪いサンプリング



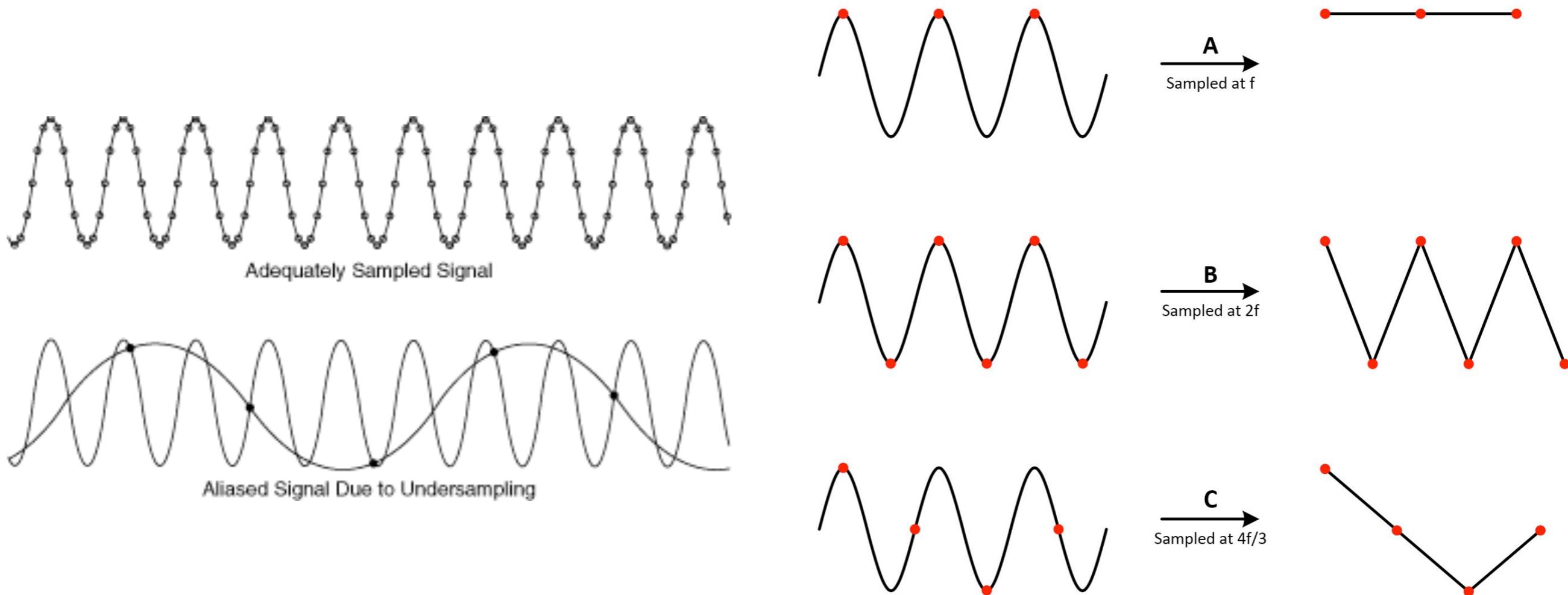
白黒の格子を再現できる良い
サンプリング



白黒の格子を再現できない悪い
サンプリング

格子の像を再現するためにはどのようなサンプリングをすればよいか。丸はサンプリング点を表す。

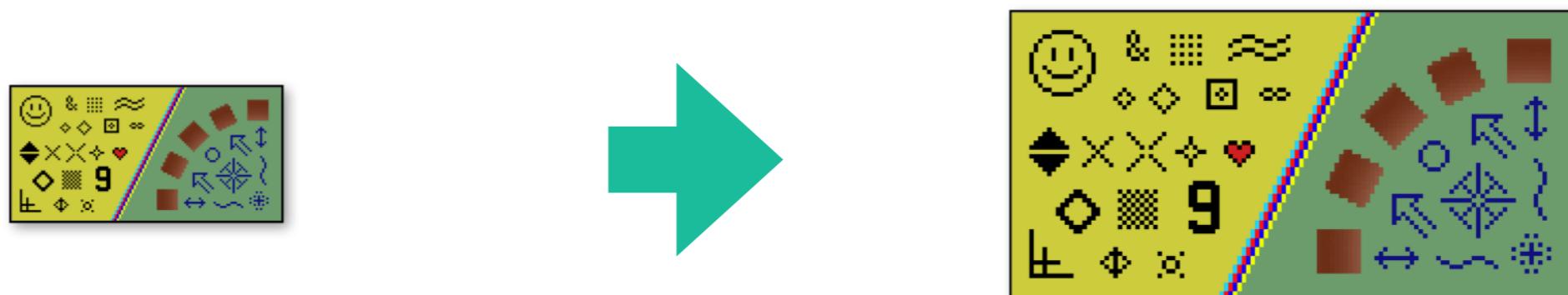
■ 良いサンプリングと悪いサンプリング



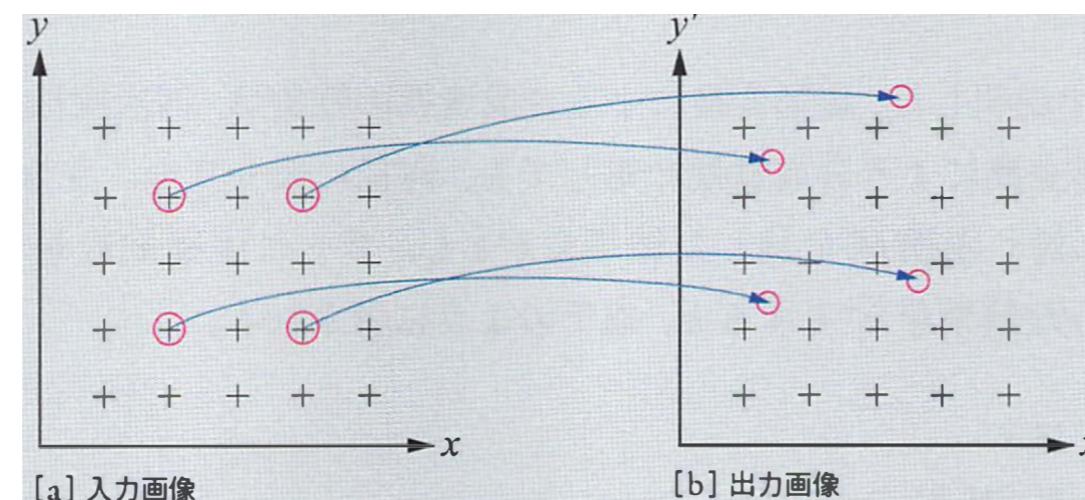
単純に画素を間引くだでは、異なる波形になることがある(Wagon-wheel effect).

■ 画像拡大・変換の問題

- 存在しない画素の画素値を推定し、綺麗に拡大できるか。



- 画素の座標は整数値であるが、変換後の座標は整数ではなくなる。そのため、整数値の座標上の画素値を決める必要がある。



■ リサンプリング

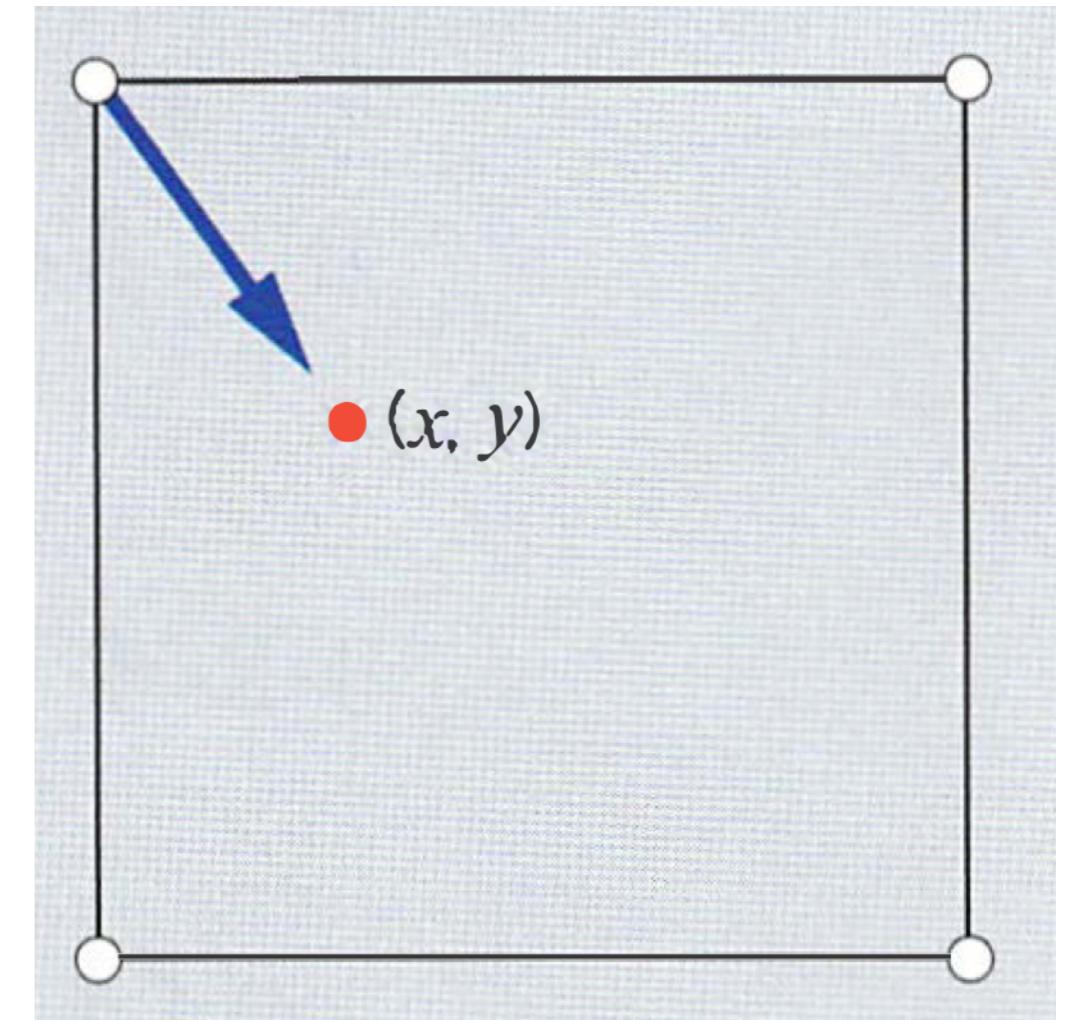
- ・画像を拡大縮小等の変換を行った場合, 画素の不足や画素の位置が小数になる問題がある. そのため, 新たに画素値を求める必要がある. これをリサンプリング(再標本化)と言う.
- ・しかし, 整数点上の画素値がない場合は, 画素値を補間する必要がある.

■ nearest neighbour

- 画素値を求める位置に最も近い画素の画素値を用いる。

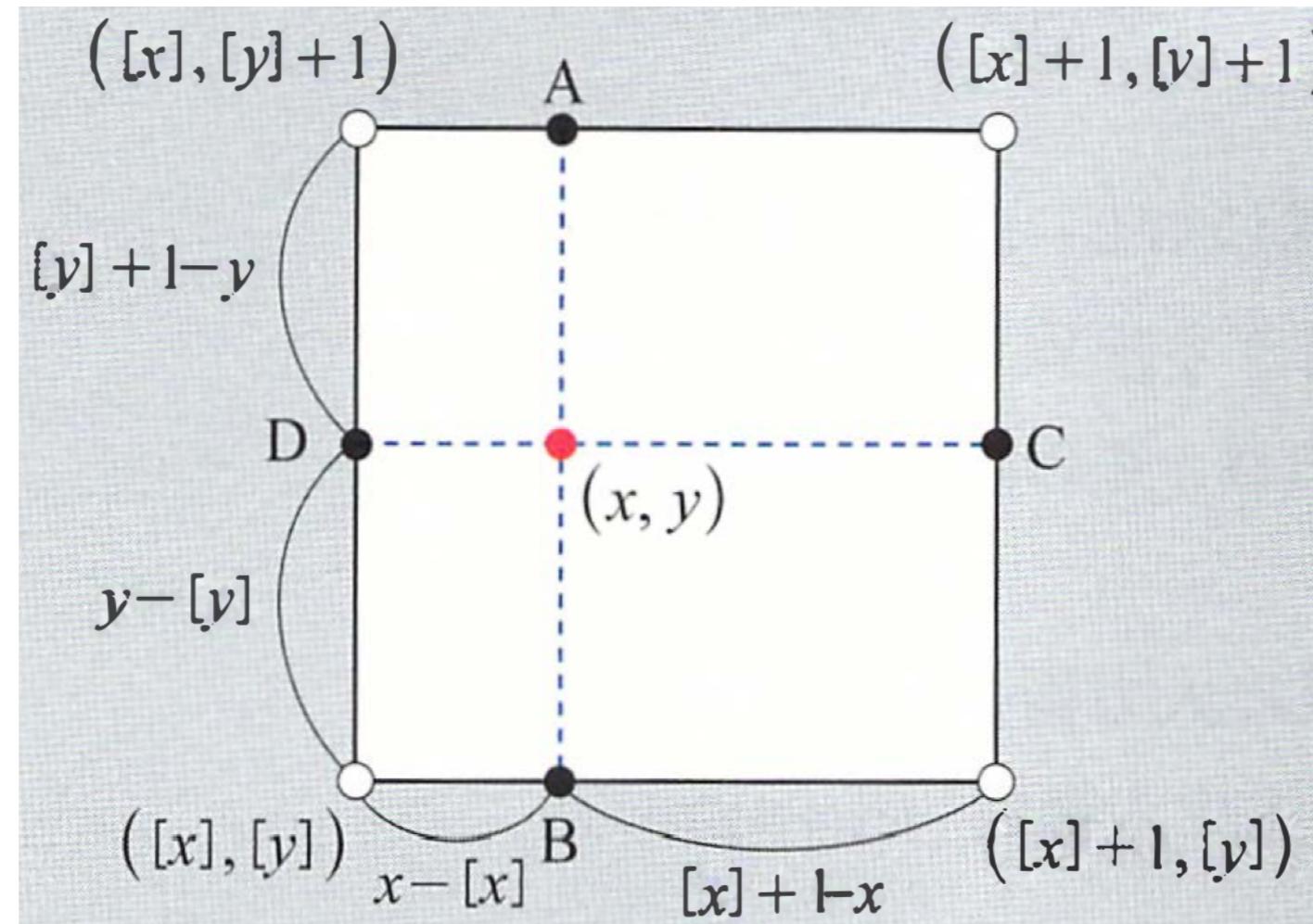
$$I(x, y) = f(\lfloor x + 0.5 \rfloor, \lfloor y + 0.5 \rfloor)$$

$I(x, y)$: 位置 (x, y) における画素値
 $f(x, y)$: 位置 (x, y) における入力画像の画素値
 $\lfloor x \rfloor$: x を超えない最大の整数(床関数)



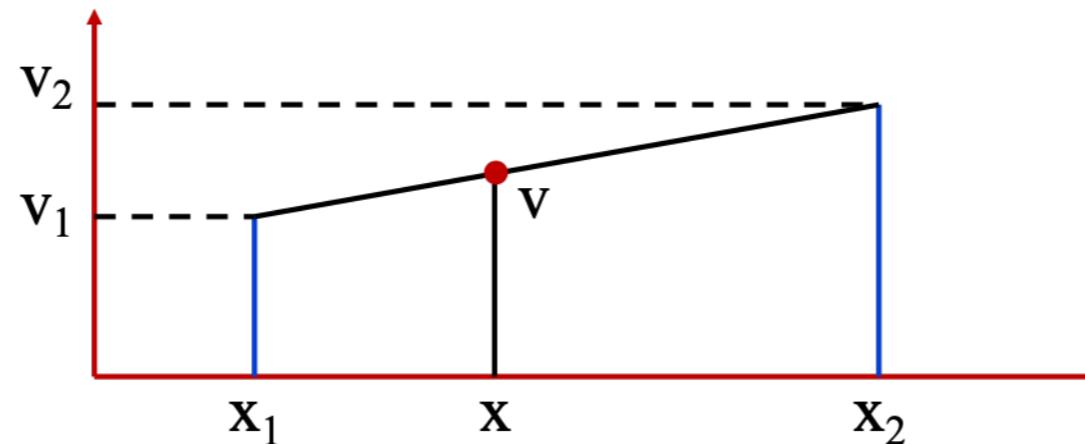
bilinear interpolation

- 周囲4画素の画素値から線形補間により位置(x, y)の画素値を求める。
- 高速で比較的良い結果が得られる



線形補間

1次式により値を補間する。

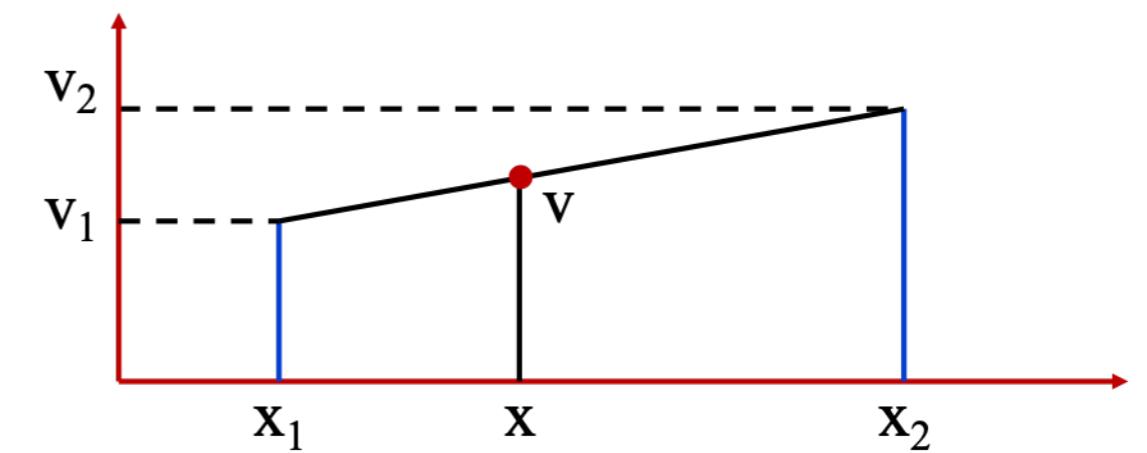


線形補間ににより v の値を求めよ。

線形補間

$$\frac{v - v_1}{x - x_1} = \frac{v_2 - v_1}{x_2 - x_1}$$

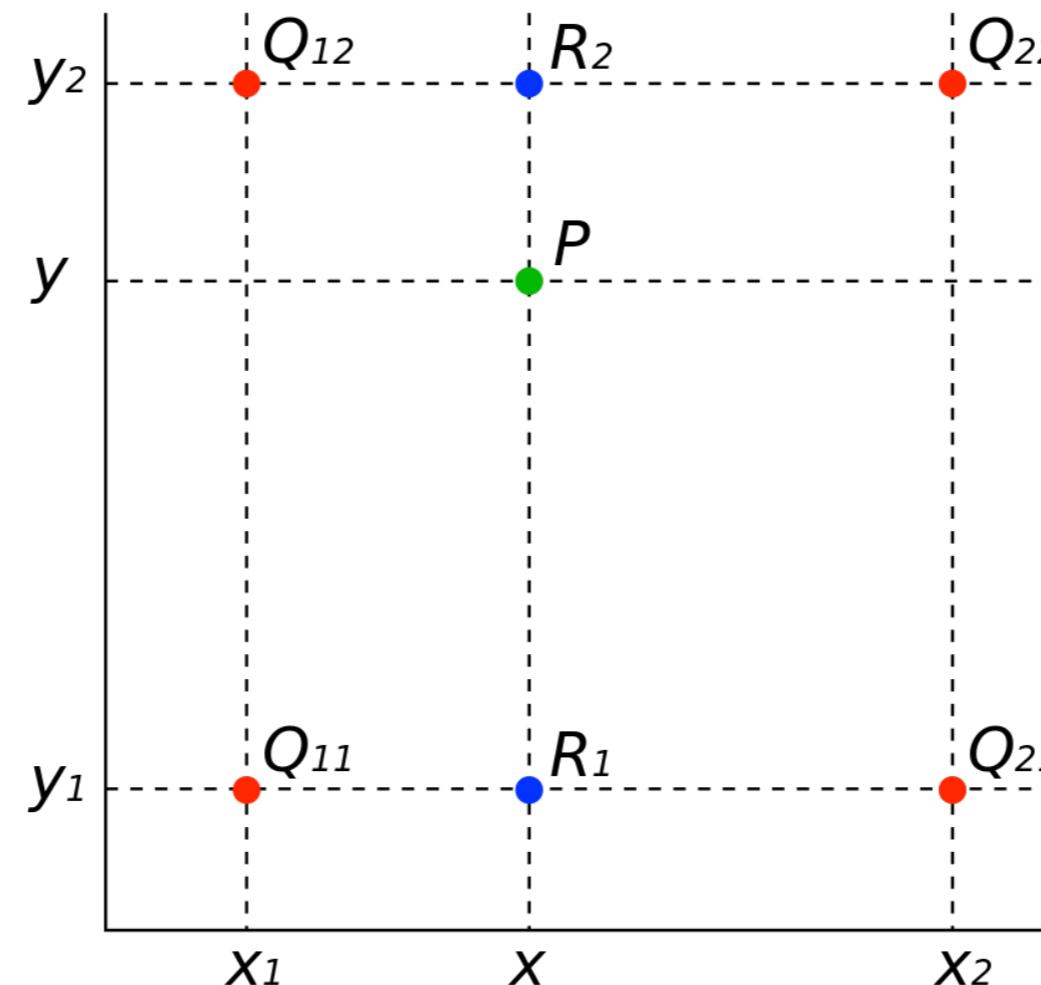
$$v = \frac{v_2 - v_1}{x_2 - x_1} (x - x_1) + v_1$$



小学校で習う比率の問題

■ 2次元座標の線形補間

- まず、点 R_1, R_2 の値を $f(R_1), f(R_2)$ を線形補間ににより求める。
- 次に、点 R_1, R_2 の値 $f(R_1), f(R_2)$ から点 P の値 $f(P)$ を線形補間ににより求める。
- 以上の作業で求まった式を用いた補間をバイリニア補間と呼ぶ。



■ バイリニア補間の式

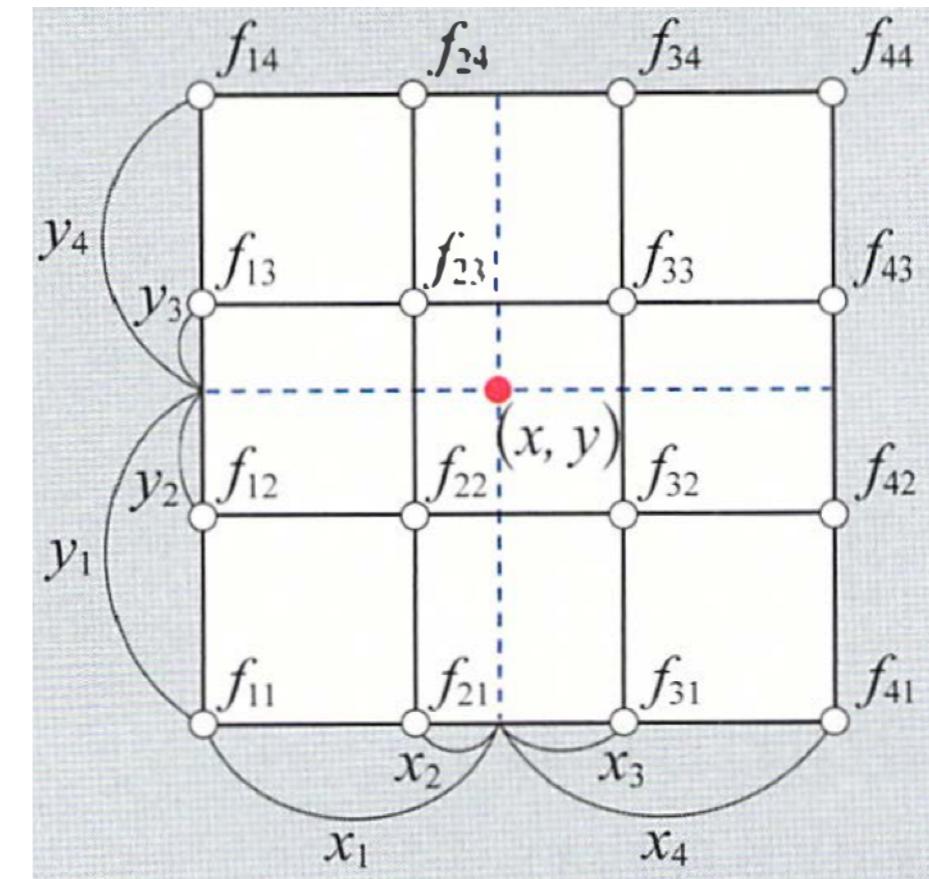
$$f(x, y) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x & x - x_1 \end{bmatrix} \begin{bmatrix} f(Q11) & f(Q12) \\ f(Q21) & f(Q22) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}$$

$f(Q11)$ は点Q11における輝度値

導出はレポートに

bicubic interpolation

周囲16画素の値を用い輝度値の補間を行う。



$$I(x, y) = (h(x_1), h(x_2), h(x_3), h(x_4)) \begin{pmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \\ f_{41} & f_{42} & f_{43} & f_{44} \end{pmatrix} \begin{pmatrix} h(y_1) \\ h(y_2) \\ h(y_3) \\ h(y_4) \end{pmatrix}$$

$$x_1 = 1 + x - [x]$$

$$x_2 = x - [x]$$

$$x_3 = [x] + 1 - x$$

$$x_4 = [x] + 2 - x$$

$$y_1 = 1 + y - [y]$$

$$y_2 = y - [y]$$

$$y_3 = [y] + 1 - y$$

$$y_4 = [y] + 2 - y$$

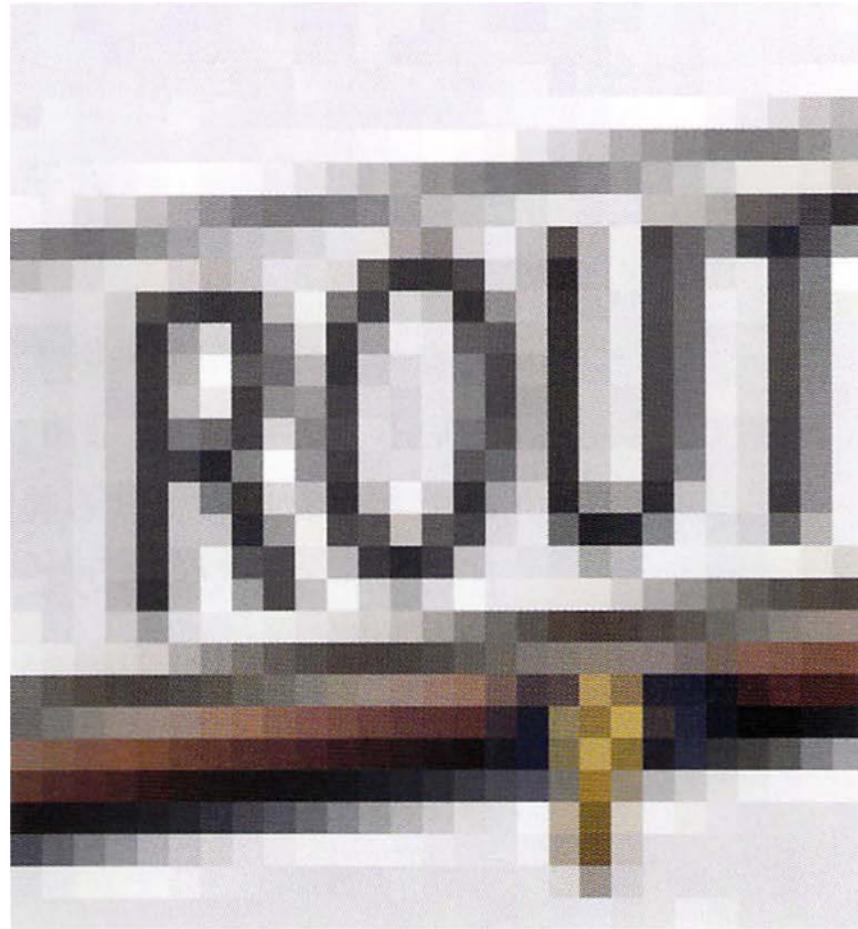
$$h(t) = \begin{cases} |t|^3 - 2|t|^2 + 1 & (|t| \leq 1) \\ -|t|^3 + 5|t|^2 - 8|t| + 4 & (1 < |t| \leq 2) \\ 0 & (2 < |t|) \end{cases}$$

なぜこうなるかは省略

■ 拡大結果の比較



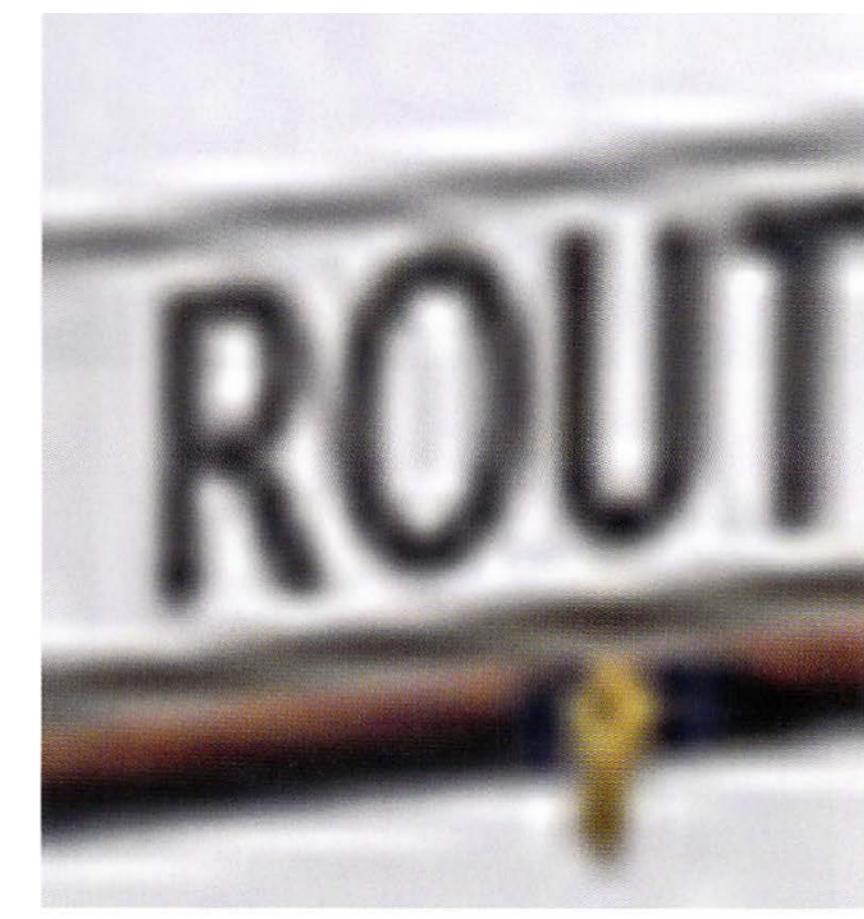
[a] 原画像



[b] ニアレストネイバーによる拡大画像

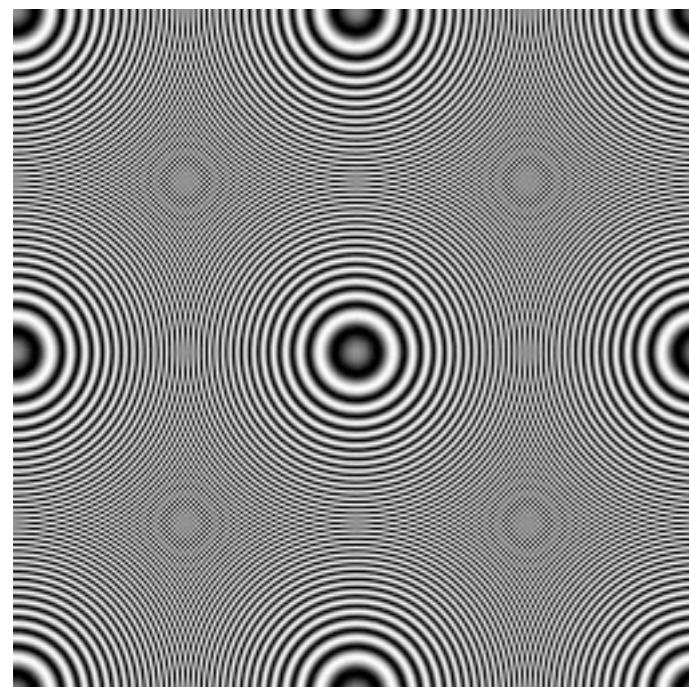


[c] バイリニア補間による拡大画像



[d] バイキュービック補間による拡大画像

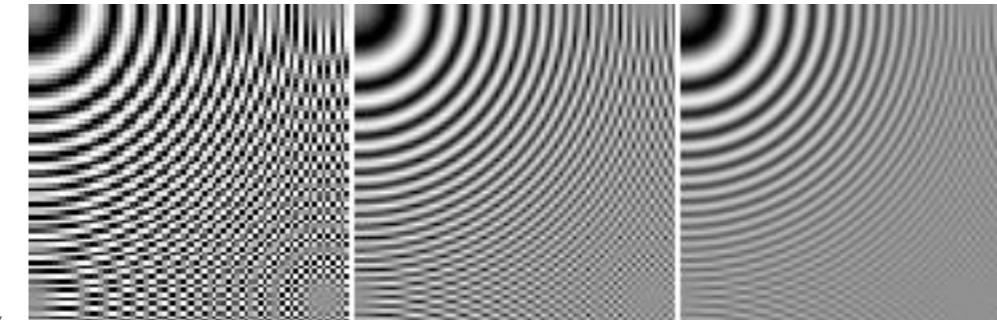
■ 拡大縮小の結果比較



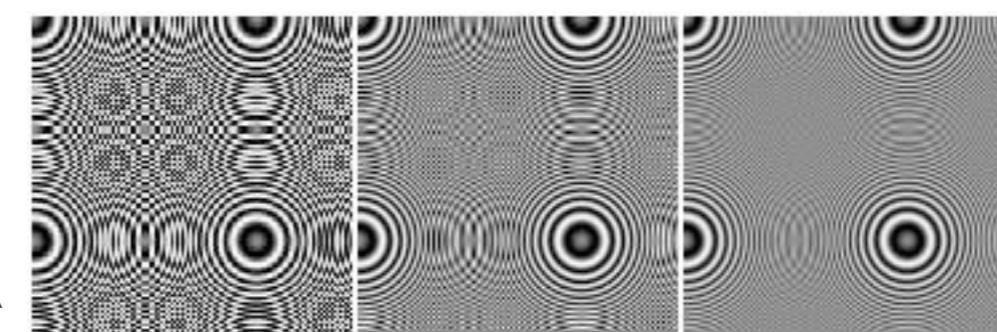
170%

65%

Nearest neighbour Bilinear Bicubic



Nearest neighbour Bilinear Bicubic



どの手法を使うかは、対象となる
画像や適用分野で決まる。