デジタルメディア処理2

担当: 井尻 敬

デジタルメディア処理２、2017（前期）

~~4/13 デジタル画像とは ：イントロダクション~~
~~4/20 フィルタ処理1 ：画素ごとの濃淡変換、線形フィルタ，非線形フィルタ~~
~~4/27 フィルタ処理2 ：フーリエ変換，ローパスフィルタ，ハイパスフィルタ~~
~~5/11 画像の幾何変換１ ：アファイン変換~~
5/18 画像の幾何変換２ ：画像の補間，イメージモザイキング
5/25 画像領域分割 ：領域拡張法，動的輪郭モデル，グラフカット法，
6/01 **前半のまとめ (約30分)と中間試験（約70分）**
6/08 特徴検出1 ：テンプレートマッチング、コーナー・エッジ検出
6/15 特徴検出2 ：DoG特徴量、SIFT特徴量、ハフ変換
6/22 画像認識1 ：パターン認識概論，サポートベクタマシン
6/29 画像認識2 ：ニューラルネットワーク、深層学習
7/06 画像処理演習 ：ImageJを使った画像処理
7/13 画像処理演習 ：Pythonプログラミング
7/20 **後半のまとめ (約30分)と期末試験（約70分）**

Contents

- 画像の変換
- 画像補間
- イメージモザイキング（パノラマ合成）

復習

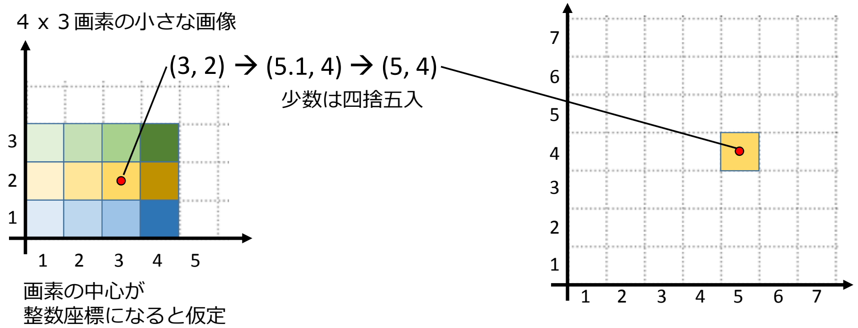
- 行列演算
- 同次形式
- アファイン変換

※先週の内容は、画像処理/CG/VRなどの分野で必須です．もし怪しい部分があったら時間をかけて復習してください

画像の変換

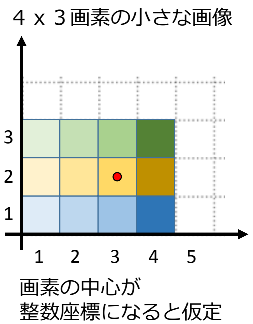
画像の変換

X軸方向に1.7倍，Y軸方向に2倍に拡大する変換を考える
画素の幅を1とする
各画素を変換先に移動してみると…

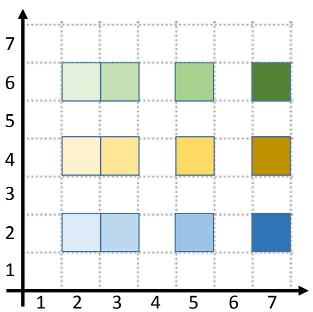


画像の変換

X軸方向に1.7倍，Y軸方向に2倍に拡大する変換を考える
画素の幅を1とする
各画素を変換先に移動してみると…

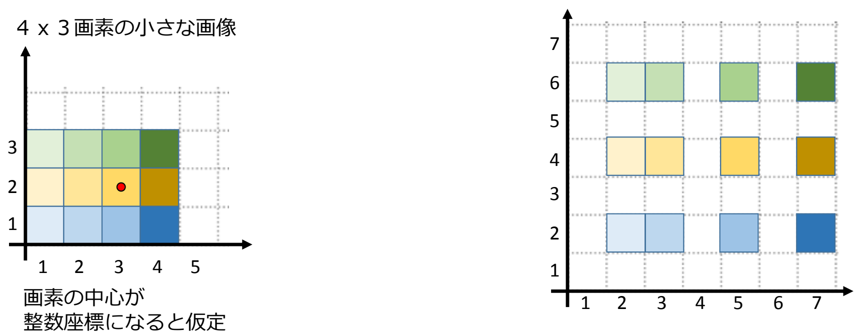


(1, 1)→(2,2)
(1, 2)→(2,4)
(1, 3)→(2,6)
(2, 1)→(3,2)
(2, 2)→(3,4)
:
:
(4, 1)→(7,2)
(4, 2)→(7,4)
(4, 3)→(7,6)



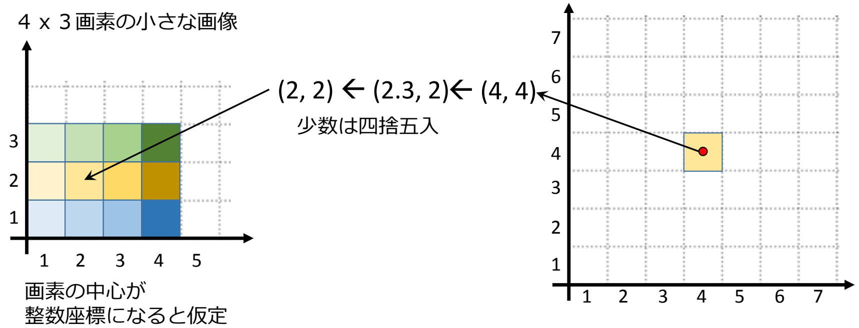
画像の変換

各画素を変換先に移動すると、飛び飛びの画像ができてしまう（拡大時）
ほしかったのはもっと密な画像…
そこで、通常は逆変換を考えます！



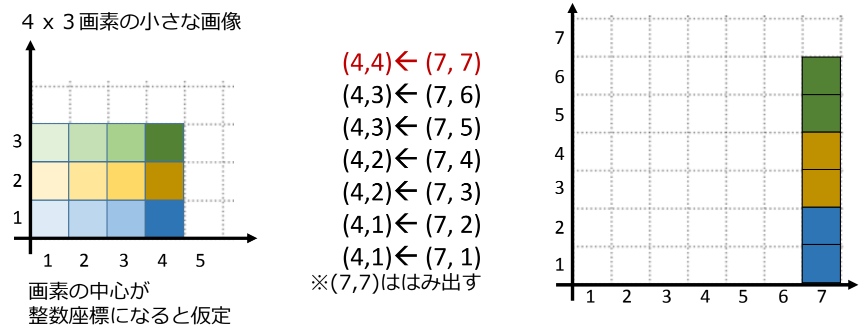
画像の変換

所望の変換は，X軸方向に1.7倍，Y軸方向に2倍
この逆変換は，X軸方向に1/1.7倍，Y軸方向に1/2倍
変換後画像の各画素に逆変換を施し，元画像における画素位置を取得する



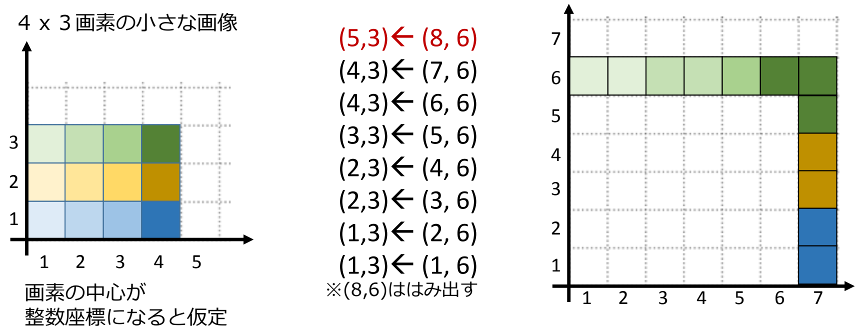
画像の変換

所望の変換は，X軸方向に1.7倍，Y軸方向に2倍
この逆変換は，X軸方向に1/1.7倍，Y軸方向に1/2倍
変換後画像の各画素に逆変換を施し，元画像における画素位置を取得する



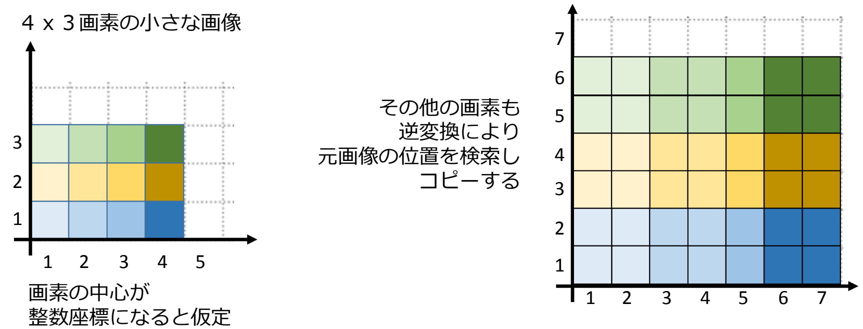
画像の変換

所望の変換は，X軸方向に1.7倍，Y軸方向に2倍
この逆変換は，X軸方向に1/1.7倍，Y軸方向に1/2倍
変換後画像の各画素に逆変換を施し，元画像における画素位置を取得する



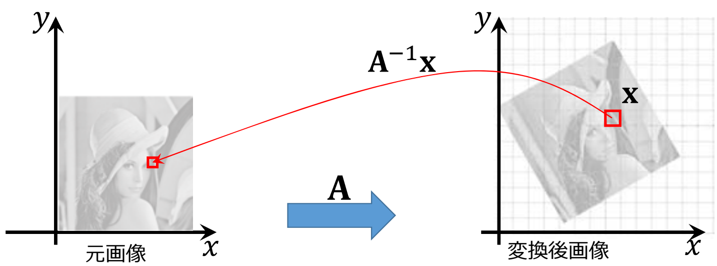
画像の変換

所望の変換は，X軸方向に1.7倍，Y軸方向に2倍
この逆変換は，X軸方向に1/1.7倍，Y軸方向に1/2倍
変換後画像の各画素に逆変換を施し，元画像における画素位置を取得する



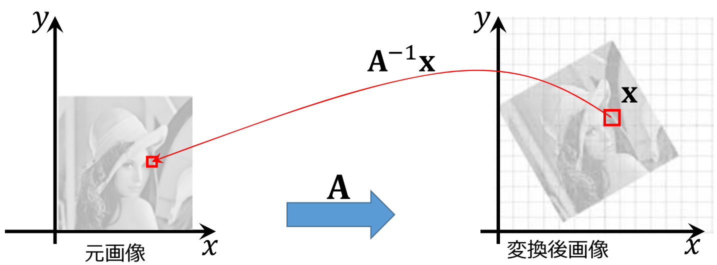
画像の変換

任意の変換について
誤）変換元画像の各画素の行き先を計算する
正）変換先の各画素に逆変換を掛け，元画像を参照する
※ X軸方向に0倍のような変換をすると逆変換が存在しないので注意



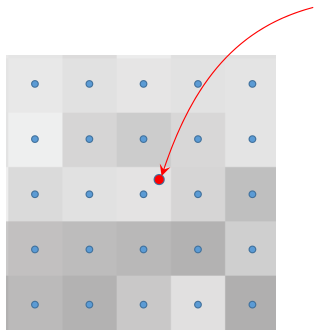
画像の補間

なぜ画像補間が必要か？



- 画像変換時には、逆変換を計算し元画像の画素を参照する
- 参照先を拡大してみると。。。

なぜ画像補間が必要か？

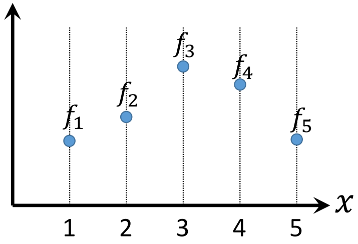


- 赤点: サンプリングしたい場所
- 青点: 画素値が存在する場所

赤点の場所の画素値は？
- 一番近い画素値を使う??
- 近傍画素を混ぜる？？

→ 補間する

補間法（1D）

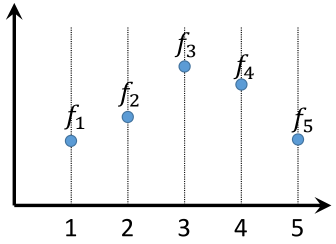


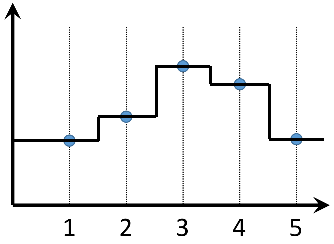
入力：画素値 f_i
xが整数の位置のみに値が存在

出力：f_iを補間した関数$g(x)$

※修正: 連続関数$g(x)$ → 関数$g(x)$

補間法（1D）：Nearest Neighbor





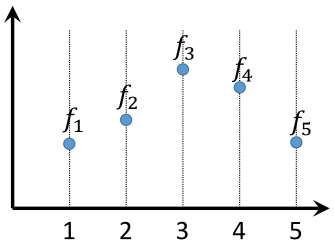
入力：画素値 f_i

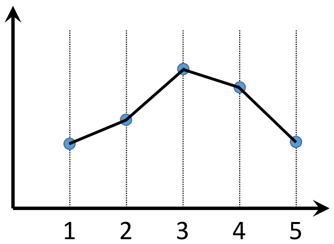
最近傍画素の値を使う

$g(x) = f_{\lfloor x+0.5 \rfloor}$

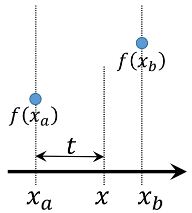
※$\lfloor t \rfloor$はガウス記号: tを超えない最大の整数

補間法（1D）：Linear Interpolation





入力：画素値 f_i



前後 2 画素を線形に補間する

$$g(x) = (1-t)f_{x_a} + tf_{x_b}$$

$$x_a = \lfloor x \rfloor,$$

$$x_b = \lfloor x \rfloor + 1,$$

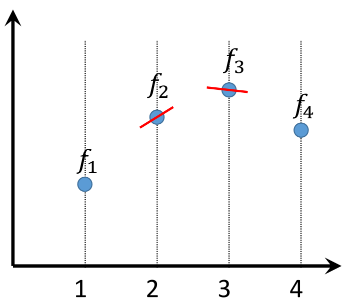
$$t = x - \lfloor x \rfloor$$

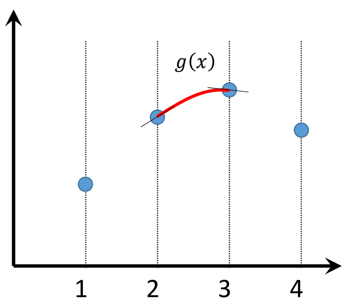
補間法（1D）：**Hermite** Cubic Spline **Interpolation**

※赤字は講義後修正したもの

区間[2,3]を補間するとき

- f_2, f_3における勾配も制約する
- 勾配制約を計算するためf_1, f_2, f_3, f_4を利用する



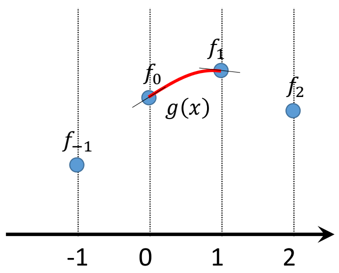


補間法（1D）：**Hermite** Cubic Spline **Interpolation**

※赤字は講義後修正したもの

入力：画素値f_{-1}, f_0, f_1, f_2

下図の区間[0,1]の補間を考える



$g(x)$は3次の関数であるとする，

$$g(x) = ax^3 + bx^2 + cx + d \qquad for\ x \in [0,1] \qquad \dots(1)$$

境界において画素値を満たすため，

$$g(0) = f_0,\ \ g(1) = f_1 \qquad \dots(2)$$

境界における勾配を４点を用いて指定

$$g'(0) = \frac{1}{2}(f_1 - f_{-1}),\ \ g'(1) = \frac{1}{2}(f_2 - f_0) \qquad \dots(3)$$

式(1)(2)(3)より$g(x)$が求まる

$$g(x) = \frac{1}{2}(1 \quad x \quad x^2 \quad x^3)\begin{pmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{pmatrix}\begin{pmatrix} f_{-1} \\ f_0 \\ f_1 \\ f_2 \end{pmatrix}$$

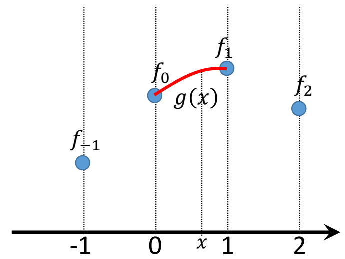
補間法（1D）：Cubic Convolution Interpolation [1]

※赤字は講義後修正したもの

教科書で紹介されているのはこれ

下図の区間[0,1]の補間を考える

$x = -1,0,1,2$の画素値をf_{-1}, f_0, f_1, f_2とする



$g(x)$を４つの画素値の重み付け和で表現する

$$g(s) = h(t_{-1})f_{-1} + h(t_0)f_0 + h(t_1)f_1 + h(t_2)f_2$$

ただし，t_i は，xから画素までの距離

$$t_{-1} = x + 1,\ \ t_0 = x,\ \ t_1 = 1 - x,\ \ t_2 = 2 - x$$

重み関数は以下の通り定義される[1]

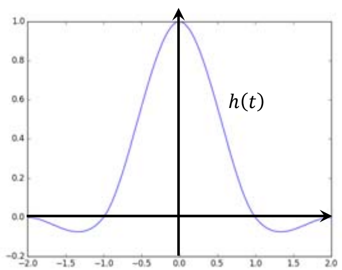
$$h(t) = \begin{cases} (a+2)|t|^3 - (a+3)|t|^2 + 1 & if\ \ |t| \le 1 \\ a|t|^3 - 5a|t|^2 + 8a|t| - 4a & if\ \ 1 \le |t| \le 2 \\ 0 & otherwise \end{cases}$$

aはユーザが決める変数，a=-0.5とするとよい[1]

[1] R. Keys, Cubic convolution interpolation for digital image processing, IEEE TASSP 1981.

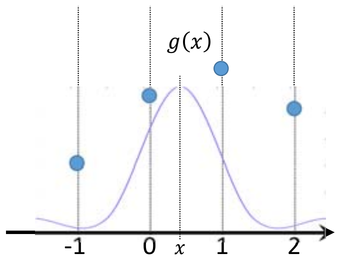
補間法（1D）：Cubic Convolution Interpolation [1]

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



h(0)=1

h(n)=0　nは0でない整数



h(x)を求めたい位置xに重ね
周囲4画素の重みを決定する

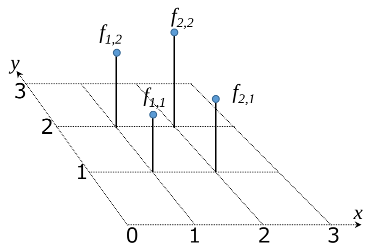
[1] R. Keys, Cubic convolution interpolation for digital image processing, IEEE TASSP 1981.

補間法（2D）

解説した各手法を2次元に拡張する

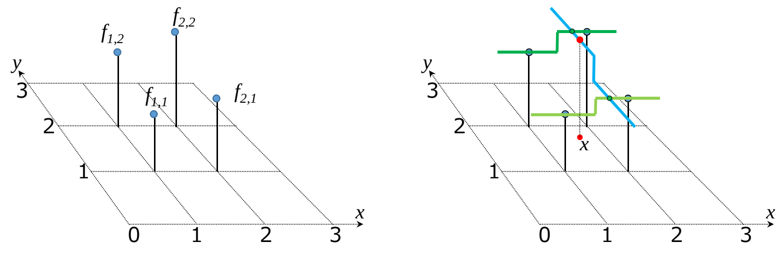
x軸方向に補間し，y軸方向に補完する

2次元補間は、bi-*という名前になる



この図では，破線の交差部分に
画素中心があるとする

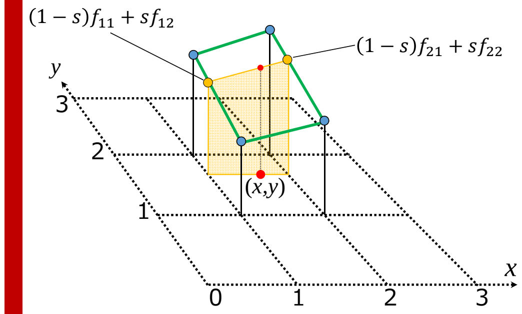
補間法（2D）：Nearest neighbor

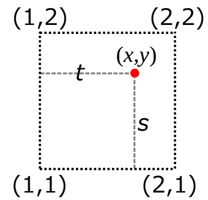


最近傍画素値を利用する

$$g(x) = f_{\lfloor x+0.5 \rfloor, \lfloor y+0.5 \rfloor}$$

補間法（2D）：Linear Interpolation





$x \in [1,2], y \in [1,2]$の範囲を

画素$f_{11}, f_{12}, f_{21}, f_{22}$より補間する

$$g(x,y) = (1-t \quad t)\begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{pmatrix}\begin{pmatrix} 1-s \\ s \end{pmatrix}$$

$$t = x-1, s = y-1$$

上式はなにをしてるのか？

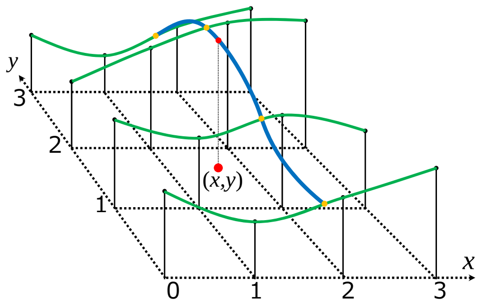
1. まずx=1, x=2においてy軸方向に線形補間し2点を取得（黄点）

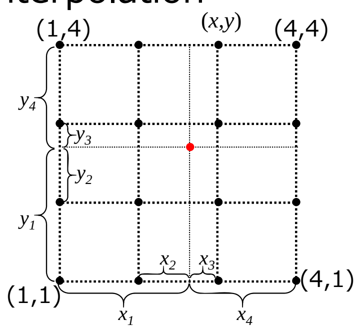
   $(1-s)f_{11} + sf_{12}$, $(1-s)f_{21} + sf_{22}$

2. 得られた2点をx軸方向に線形補間（赤点）

   $(1-t)\big((1-s)f_{11} + sf_{12}\big) + t\big((1-s)f_{21} + sf_{22}\big)$

補間法（2D）：Bicubic Convolution Interpolation





$x \in [1,2], y \in [1,2]$の範囲を近傍16画素f_{xy}より補間する

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

$h(t)$は1次元補間と同様, x_i, y_iは右上図の通り定義される.

左上図の通り

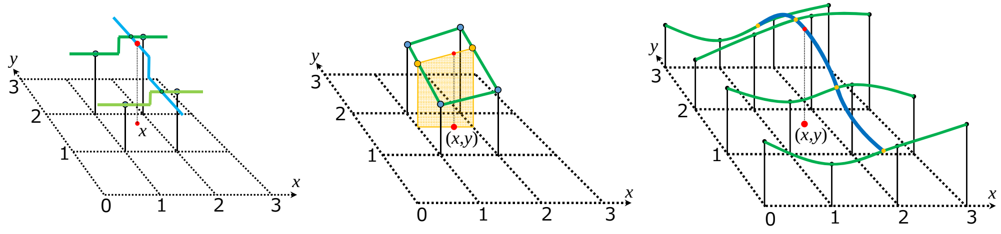
1. まずx軸に沿ってcubic補間
2. 得られた4点を利用しy軸に沿ってcubic補間

画像の補間法：例

教科書p171の図参照

まとめ: 画像の補間法

- 画像の変換（特に拡大）の際，画像の画素と画素の間を参照する
- 周囲の画素を利用し，参照位置の画素値を決定する



Nearest Neighbor　　Bilinear Interpolation　　Bicubic Interpolation

- 様々なソフトウエアがこの変換(Bicubicが多い)を自動でかけてくれる
- **研究目的のデータ処理においては注意**が必要

→ デモ VoTraver volume rendering

イメージモザイキング（パノラマ合成）

イメージモザイキング

panorama.py

- ここまで紹介してきた画像変換の応用のひとつ
- 複数の画像を変形し重ね合わせて大きな画像を作成する技術

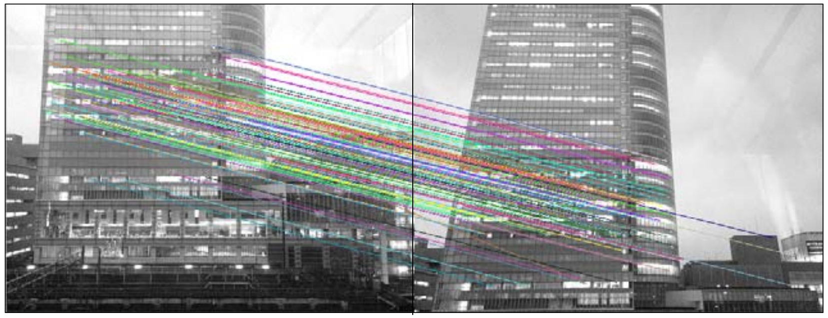
パノラマ合成1: 入力画像について特頂点を検出する

- **特徴点**：角やエッジなど，顕著な局所的変化がある場所
- 特徴点検出アルゴリズムはSIFT, SURF, Eigen, Harrisなどが有名
- 特徴点は，その周囲の様子を記述する特徴ベクトルを持つ

※特徴点については後で詳しく解説

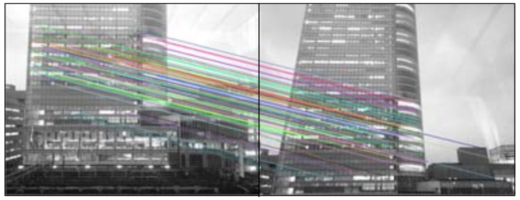
※左図はAKAZE algorithmを利用した結果

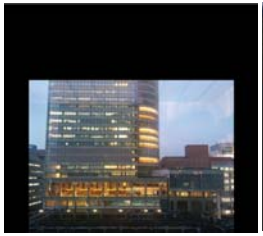
パノラマ合成2. 特徴点の対応付け



- 各特徴点は局所領域の特徴を記述する特徴ベクトルを持つ
- 特徴ベクトルの類似性を利用して対応を計算する
- 上図ではBrute force algorithmを利用
  - 左画像の特徴点をひとつピックアップし、最も似た特徴点を右画像内から全検索

パノラマ合成3. 変換行列の計算



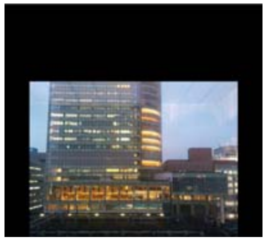


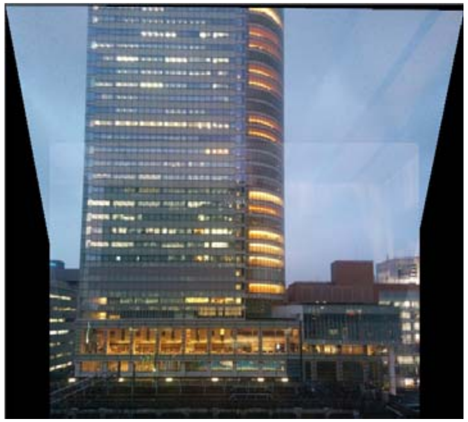
- 対応特徴点の位置が重なるよう右画像を射影変換
- つまり，対応点をなるべく一致させる行列

$$\mathbf{H} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix}$$ を求めたい

- RANSAC（Random Sample Consensus）
  1. 未知変数推定に必要なデータを乱択する (未知変数が8個なので　個の特徴点の組)
  2. 選択した特徴点の組を用いて変換$\mathbf{H}$を導出
  3. 変換$\mathbf{H}$によりほか全て特徴点を変換する
     特徴点が対応点の十分近くに変換された → Inlier
     特徴点の変換先が対応点から遠い → Outlier
  4. 1~3を繰り返しInlier数が最多の$\mathbf{H}$を出力

パノラマ合成4. 画像の合成

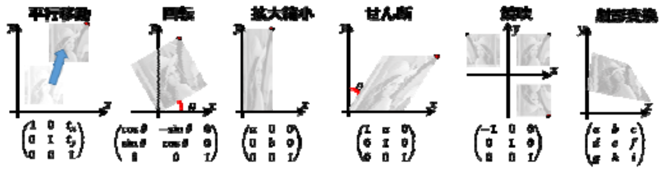




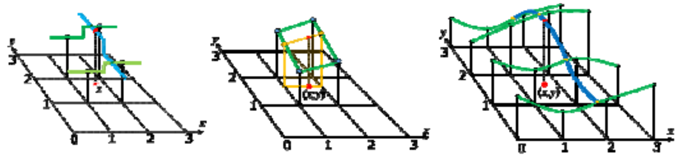
- 上図は単純な実装: 2画像が重なる部分は両者の平均を取る → シームが目立つ
- 目立たないシームを計算する手法 → [GraphCutTextures, SIGGRAPH 2003]
- 画像ピラミッドを利用する手法 → [A Multiresolution Spline With Application to Image Mosaics, TOG1983]

まとめ: 画像の変換

- **幾何学変換**を紹介した
  - Affine 変換
  - 射影変換
  - 同次座標系



- **補間法**を紹介した
  - Nearest Neighbor
  - Bilinear Interpolation
  - Bicubic Interpolation



- **パノラマ合成**を紹介した