情報デザイン専攻

画像情報処理論及び演習I

-デジタル画像の表現と応用-
その2：アフィン変換と画素値の補間
補足資料

第５回講義
水曜日１限
教室6218情報処理実習室

吉澤　信
shin@riken.jp, 非常勤講師
大妻女子大学　社会情報学部
独立行政法人 理化学研究所

Shin Yoshizawa: shin@riken.jp

今日の授業内容

www.riken.jp/brict/Yoshizawa/Lectures/index.html

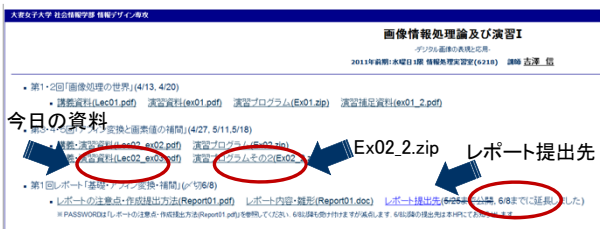
① 復習、順変換、逆変換と補間法
② レポートの説明
③ 演習の説明
④ 演習：アフィン変換

Shin Yoshizawa: shin@riken.jp

復習：講義・演習資料・準備

www.riken.jp/brict/Yoshizawa/Lectures/index.html

✓ Ex02.zipの代わりにEx02_2.zipを用意しました。



✓ 第一回レポートの〆切を6月8日までに延長しました。

Shin Yoshizawa: shin@riken.jp

復習：アフィン変換とは？

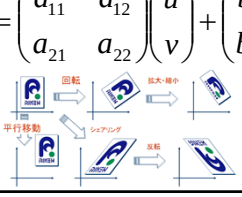
✓ Affine Transformationは既知の行列 A とベクトル $\mathbf{t}$ を用いて**線形変換** $\mathbf{y} = f(\mathbf{x}) = A\mathbf{x} + \mathbf{t}$ により点 $\mathbf{x}$ を点 $\mathbf{y}$ に写像する.

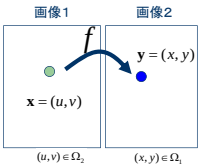
✓ 2次元では、

$$\mathbf{y} = (x, y), \mathbf{x} = (u, v), \mathbf{t} = (b_1, b_2), A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix},$$

とすると、アフィン変換は以下の様に書ける.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

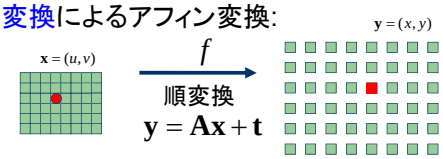




Shin Yoshizawa: shin@riken.jp

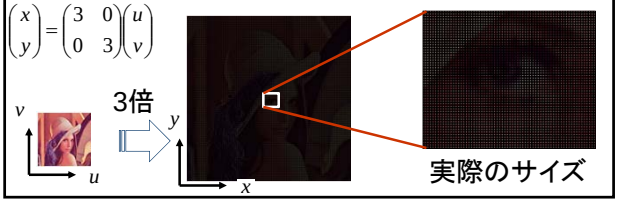
実際に画像のアフィン変換をしてみると…

✓ 順変換によるアフィン変換:



✓ 例えば3倍に拡大すると…

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

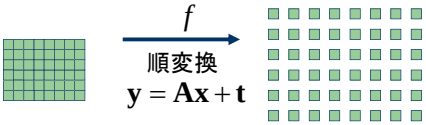


Shin Yoshizawa: shin@riken.jp

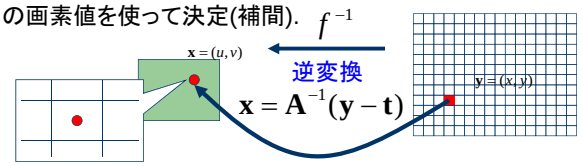
重要：順変換と逆変換１

✓ 逆変換 ＋ 画素値の補間によるアフィン変換:

- 順写像(Forward Mapping) VS 逆写像(Backward Mapping):

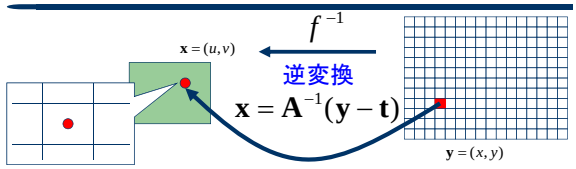


逆変換した位置の画素値を周り
の画素値を使って決定(補間).



Shin Yoshizawa: shin@riken.jp

順変換と逆変換2



$$\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{pmatrix}\begin{pmatrix}u\\v\end{pmatrix}+\begin{pmatrix}b_1\\b_2\end{pmatrix}$$

✓ ad−bc=0になる様な変換は作ってはいけない.

✓ ↓なので、

$$A=\begin{pmatrix}a&b\\c&d\end{pmatrix}\Leftrightarrow A^{-1}=\frac{1}{ad-bc}\begin{pmatrix}d&-b\\-c&a\end{pmatrix}$$

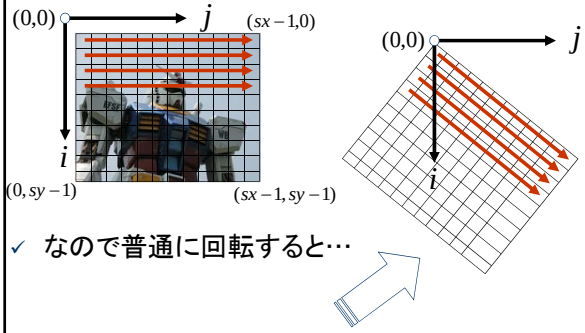
逆変換:

$$\begin{pmatrix}u\\v\end{pmatrix}=\frac{1}{a_{11}a_{22}-a_{12}a_{21}}\begin{pmatrix}a_{22}&-a_{12}\\-a_{21}&a_{11}\end{pmatrix}\begin{pmatrix}x-b_1\\y-b_2\end{pmatrix}$$

Shin Yoshizawa: shin@riken.jp

復習:回転:Rotation

✓ 画像の座標系は…



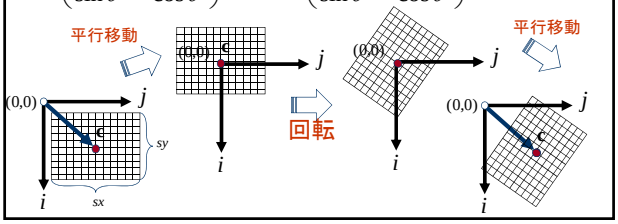
✓ なので普通に回転すると…

Shin Yoshizawa: shin@riken.jp

復習:回転:Rotation

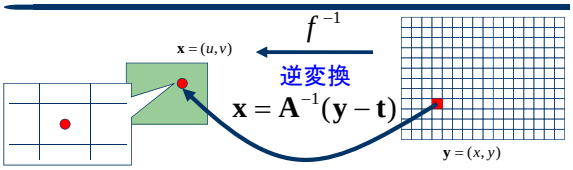
✓ 画像の中心を原点とする座標系で回転させたい場合は一回、中心に平行移動させて回転後に逆向きの平行移動をする: $\mathbf{c}=(sx/2,sy/2)$

$$\mathbf{y}=\begin{pmatrix}\cos\theta&-\sin\theta\\\sin\theta&\cos\theta\end{pmatrix}\mathbf{x}\Rightarrow\mathbf{y}=\begin{pmatrix}\cos\theta&-\sin\theta\\\sin\theta&\cos\theta\end{pmatrix}(\mathbf{x}-\mathbf{c})+\mathbf{c}$$



Shin Yoshizawa: shin@riken.jp

順変換と逆変換3



画像の中心を原点とする座標系では:

$$\mathbf{x}=\mathbf{A}^{-1}(\mathbf{y}-\mathbf{c}-\mathbf{t})+\mathbf{c},\quad \mathbf{c}=(c_x,c_y)=(sx/2,sy/2),$$

逆変換:

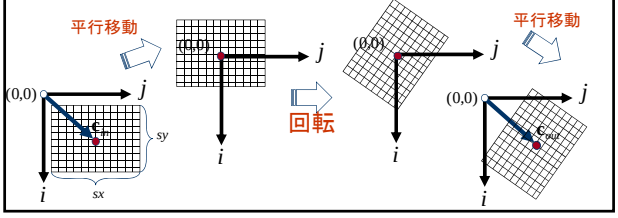
$$\begin{pmatrix}u\\v\end{pmatrix}=\frac{1}{a_{11}a_{22}-a_{12}a_{21}}\begin{pmatrix}a_{22}&-a_{12}\\-a_{21}&a_{11}\end{pmatrix}\begin{pmatrix}x-c_x-b_1\\y-c_y-b_2\end{pmatrix}+\begin{pmatrix}c_x\\c_y\end{pmatrix}$$

Shin Yoshizawa: shin@riken.jp

復習:回転:Rotation

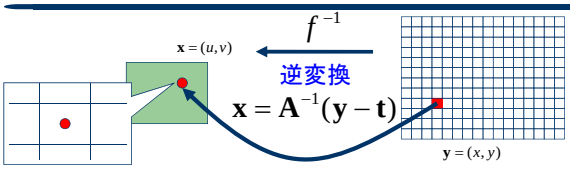
✓ 入力と出力の画像サイズが違うとき:

$$\mathbf{y}=\begin{pmatrix}\cos\theta&-\sin\theta\\\sin\theta&\cos\theta\end{pmatrix}(\mathbf{x}-\mathbf{c}_{in})+\mathbf{c}_{out}$$



Shin Yoshizawa: shin@riken.jp

順変換と逆変換4



✓ 入力と出力の画像サイズが違うとき:

$$\mathbf{x}=\mathbf{A}^{-1}(\mathbf{y}-\mathbf{c}_{out}-\mathbf{t})+\mathbf{c}_{in},$$

逆変換:

$$\begin{pmatrix}u\\v\end{pmatrix}=\frac{1}{a_{11}a_{22}-a_{12}a_{21}}\begin{pmatrix}a_{22}&-a_{12}\\-a_{21}&a_{11}\end{pmatrix}\begin{pmatrix}x-c^{out}{}_x-b_1\\y-c^{out}{}_y-b_2\end{pmatrix}+\begin{pmatrix}c^{in}{}_x\\c^{in}{}_y\end{pmatrix}$$

Shin Yoshizawa: shin@riken.jp

アフィン変換の実装

✓ 中心で逆変換をする関数を作ってみよう！

$$\mathbf{x} = \mathbf{A}^{-1}(\mathbf{y} - \mathbf{c}_{out} - \mathbf{t}) + \mathbf{c}_{in},$$

```
void AffineTransformation_BW(double inX[2],double A[2][2],
 double t[2], double ci[2],double co[2], double outX[2]){
    double tmp1[2],tmp2[2],tmp3[2];
    double mt[2]; mt[0] = -(co[0]+t[0]); mt[1] = -(co[1]+t[1]);
   Translate(inX,mt,tmp1);
    Vector_Matrix_Multiplication_Inverse(tmp1,A,tmp2);
    Translate(tmp2,ci,outX);
}
```

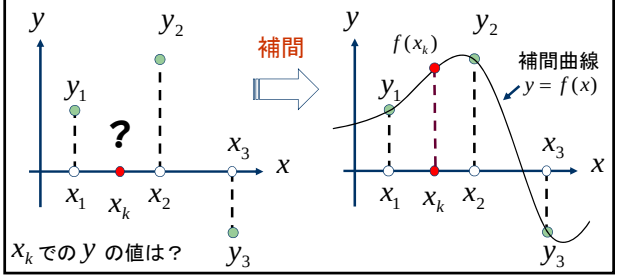
逆行列を計算する関数.

Ex02_2.zip内のex02_0.cxxに前回までも含めて既に書いてあります。

Shin Yoshizawa: shin@riken.jp

補間(Interpolation)とは？

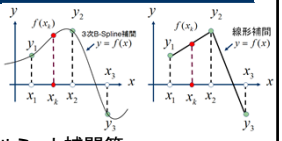
✓ 与えられた既知の数値データ列を基に、そのデータ列の間の数値(又は既知の数値データ列を通る関数)を求める事、内挿とも言う. 与えられたデータ外を考える場合は補外(外挿)と言う.



Shin Yoshizawa: shin@riken.jp

様々な補間法

✓ 多くの方法がある:
- ラグランジュ補間、多項式、
- 線形補間(直線の式)、Sinc関数、
- スプライン補間(B-Spline, etc.)、
- RBF (Radial Basis Function)、エルミート補間等.



✓ 一番基本でよく用いられているのは線形補間と3次スプライン補間(Cubic Spline Interpolation).

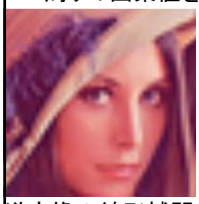
✓ 補間はいろいろ使える.

©Y. Ohtake, 2011. ©S. Yoshizawa et al. ACM SMA, 2003.

Shin Yoshizawa: shin@riken.jp

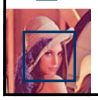
重要:画像では？画素値の補間(2D)

✓ 周りの画素値を使って補間:

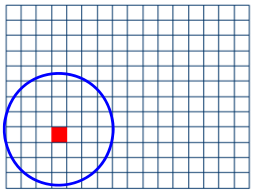


逆変換＋線形補間

5倍拡大



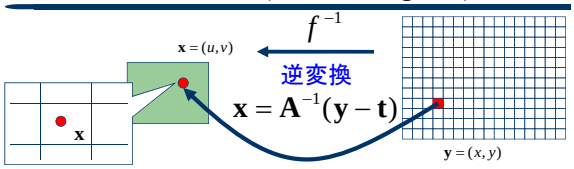
順変換



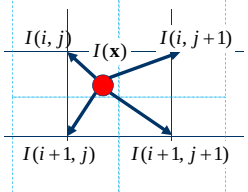
今日は…
最近傍法
線形補間法
3次Spline補間法

Shin Yoshizawa: shin@riken.jp

補間:最近傍法(Nearest Neighbor)



✓ 近傍4つの画素値のうち最も近い画素の値を使う.



$$I(\mathbf{x}) = \arg\min_{I}(|\mathbf{x} - \mathbf{p}_j|), \quad j = 0,1,2,3.$$

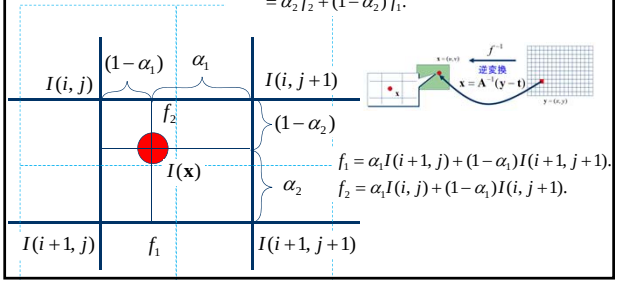
$$\mathbf{p}_0 = (i, j), \quad \mathbf{p}_1 = (i, j+1),$$
$$\mathbf{p}_2 = (i+1, j), \quad \mathbf{p}_3 = (i+1, j+1).$$

Shin Yoshizawa: shin@riken.jp

補間:線形補間法(Linear Interpolation)

✓ 近傍4つの画素値を線形補間する.

$$I(\mathbf{x}) = \alpha_2(\alpha_1 I(i, j) + (1-\alpha_1) I(i, j+1)) + (1-\alpha_2)(\alpha_1 I(i+1, j) + (1-\alpha_1) I(i+1, j+1))$$
$$= \alpha_2 f_2 + (1-\alpha_2) f_1.$$

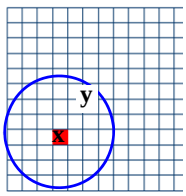


$$f_1 = \alpha_1 I(i+1, j) + (1-\alpha_1) I(i+1, j+1).$$
$$f_2 = \alpha_1 I(i, j) + (1-\alpha_1) I(i, j+1).$$

Shin Yoshizawa: shin@riken.jp

補間：線形畳み込み法

- ✓ 線形補間や3次Spline補間は「線形畳み込み(Linear Convolution)」と呼ばれる重み付和で計算出来る.重みの事を補間関数という.
- ✓ この方法は画像等の間隔が同じ格子が並んでいる場合に簡単に適用出来る.



g とIの畳み込み：

補間したい画素の位置 $\mathbf{x}$　　既知の輝度値

$$I^{\text{new}}(\mathbf{x}) = \int g(\mathbf{x}-\mathbf{y})I(\mathbf{y})d\mathbf{y}$$

補間された輝度値　　補間関数

$\mathbf{x}$近傍の画素の位置 $\mathbf{y}$

Shin Yoshizawa: shin@riken.jp

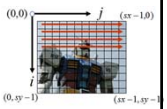
補間：線形畳み込み法

- ✓ 画像では、

$$I^{\text{new}}(\mathbf{x}) = \int g(\mathbf{x}-\mathbf{y})I(\mathbf{y})d\mathbf{y} = \iint g(x-u, y-v)I(u,v)dudv,$$

もしも、　$g(u,v) = g_1(u)g_2(v)$　ならば、

$$\begin{aligned} I^{\text{new}}(\mathbf{x}) &= \iint g(x-u, y-v)I(u,v)dudv \\ &= \iint g_1(x-u)g_2(y-v)I(u,v)dudv \\ &= \int g_2(y-v)\left(\int g_1(x-u)I(u,v)du\right)dv \\ &\approx \sum_i^N g_2(y-i)\left(\sum_j^M g_1(x-j)I[i][j]\right). \end{aligned}$$



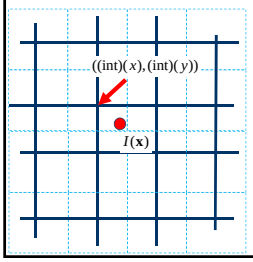
Shin Yoshizawa: shin@riken.jp

補間：3次補間法(Cubic Interpolation)

- ✓ 近傍16個の画素値を3次補間する.

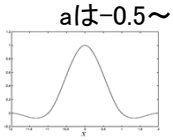
$$I^{\text{new}}(\mathbf{x}) = \sum_{i=(\text{int})(y)-1}^{(\text{int})(y)+2} g(y-i)\left(\sum_{j=(\text{int})(x)-1}^{(\text{int})(x)+2} g(x-j)I(i,j)\right).$$

$\mathbf{x}=(x,y)$

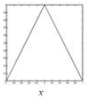


$$g(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1, & 0 \le |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a, & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases}$$

aは−0.5～2がよい.



3次補間関数



線形補間関数

Shin Yoshizawa: shin@riken.jp

重要：補間法実装

- ✓ Ex02_2/interpolation.hに(Ex02/interpolation.hと同じ)
  - 最近傍法：NearestNeighbor(…)
  - 線形補間法：LinearInterpolation(…)
  - 3次補間法：CubicInterpolation(…)

が実装されているのでよく見ておいてください.

Shin Yoshizawa: shin@riken.jp

レポートの説明

www.riken.jp/brict/Yoshizawa/Lectures/index.html

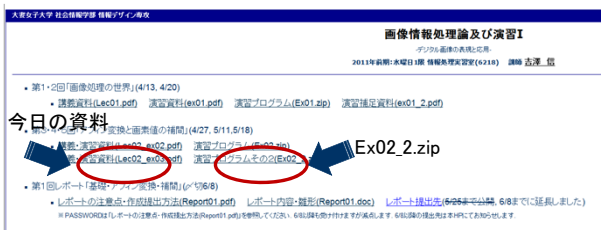
- ✓ 第一回レポートの〆切を6月8日までに延長しました。

Shin Yoshizawa: shin@riken.jp

演習の説明

www.riken.jp/brict/Yoshizawa/Lectures/index.html

- ✓ Ex02.zipの代わりにEx02_2.zipを用意しました。



Shin Yoshizawa: shin@riken.jp

重要！演習の説明

- ✓ Ex02_2.zipの内容(ソースファイルmain):
  - Ex02_0.cxx: 順変換(pgm)
  - ex02_1.cxx: 順変換(ppm).
  - ex02_2.cxx: 逆変換(ppm,補間なし)
  - ex02_3.cxx: 逆変換 (ppm,補間あり)
  - 中を見てそれぞれ動かしてみましょう！
  - レポートではpgm, ppm両方について色々なアフィン変換を実行した結果とプログラムを提出してもらうので、↑のプログラムは動かせる様にしておいてください.

Shin Yoshizawa: shin@riken.jp

今日の演習の流れ

1. Ex02_2.zipのダウンロード(補足資料).
2. Ex02_2.zipの展開(補足資料).
3. Ex02_2内のプログラムのコンパイル.
4. Ex02_2内のプログラムの実行.
5. Ex02_2内のプログラムを書き換えて保存.
6. Ex02_2内のプログラムの再コンパイル.
7. Ex02_2内のプログラムの実行.

以下5-7の繰り返し.

Shin Yoshizawa: shin@riken.jp

演習の説明: Ex02_2内のプログラムのコンパイル

- ✓ Makefileを実行:
  1. 端末を立ち上げる.
  2. Ex02_2を展開したディレクトリーを開く.
     端末にて「cd ~/学籍番号_Ex02/Ex02_2」を入力しエンターキーを押す.
  3. 端末にて「make」を入力しエンターキーを押す.
  4. コンパイルされたかの確認:端末にて「ls」を入力しエンターキーを押す→ex02_0, ex02_1, ex02_2, ex02_3と四つの実行ファイルが出来ていたら成功.

  Makefileの中を「more Makefile」や「emacs Makefile &」で見てみましょう. 前回まで端末に直接打ち込んでいたg++のコマンドが書いてあります.

Shin Yoshizawa: shin@riken.jp

演習の説明: プログラムの実行方法

Ex02_2内にあるシェルスクリプトを実行する場合:
Run_ex02_0.sh, Run_ex02_1.sh, Run_ex02_2.sh, Run_ex02_3.sh

1. 端末を立ち上げる.
2. Ex02_2を展開したディレクトリーを開く.
   端末にて「cd ~/学籍番号_Ex02/Ex02_2」を入力しエンターキーを押す.「ls」にて上記ファイルがある事を必ず確認する事！
3. 端末にて
   「./シェルスクリプトファイル名」を入力しエンターキーを押す. 例えば「./Run_ex02_1.sh」を入力してエンターキーを押す.

注意:エラーが出る場合は端末で「chmod u+x ./*.sh」と入力してエンターキーを押す. 又は「sh Run_ex02_1.sh」を入力してエンターキーを押す.

Shin Yoshizawa: shin@riken.jp

演習: 実際に逆変換をしてみる！

- ✓ Run_ex02_0.sh, Run_ex02_1.sh, Run_ex02_2.sh, Run_ex02_3.shを全部動かしてみてください。
- ✓ 拡大してみよう！:ex02_0.cxxのmain内の上の方でdouble A[2][2];にA[1][0] = 1.0…と入れている下のRotation()をコメントアウトしてScaling()のコメントアウトを外す.

```
double A[2][2];
A[0][0]=1.0; A[0][1] = 0.0;
A[1][0] = 0.0; A[1][1] = 1.0;

//Rotation(A,45.0);
Scaling(A,3.0,3.0);
```

- ✓ ex02_0.cxxをセーブして実行してみてください.

Shin Yoshizawa: shin@riken.jp

演習: 実際に逆変換をしてみる！

- ✓ ex02_0.cxxのmainの中にある/* Forward Mapping */以下の順変換をコメントアウトして、その下の逆変換のコメントアウトを外してと2倍の拡大を実行してみてください.
- ✓ ex02_1.cxx, ex02_2.cxx, ex02_3.cxxのmain内を書き換えて色々なアフィン変換を実行してみましょう！

講義・演習内容

シラバス

1回	画像とは何か。画像の構成要素、画像の表現	基礎
2回	画像とは何か。画像の構成要素、画像の表現	
3回	画像とは何か。画像の構成要素、画像の表現	
4回	色彩の表現。加法混色、減法混色、彩度等	色彩
5回	色彩の表現。加法混色、減法混色、彩度等	
6回	色彩の表現。加法混色、減法混色、彩度等	

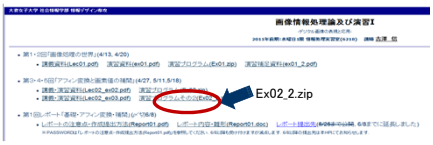
~~第1・2回：画像処理の世界~~
~~第3-5回：アフィン変換と画素値の補間~~
第6回：画像化・画像処理法・色彩

演習の補足

- ✓ Ex02_2.zipの内容(ソースファイルHeader):
  - SimpleImage.h: 画像クラス
  - pgmio.h: pgm画像(グレースケール画像)入出力用関数群.
  - pgmio.h: ppm画像(カラー画像)入出力用関数群.
  - affine.h: アフィン変換行列作成、アフィン変換座標計算の関数群.
  - interpolation.h: 補間用関数群.

演習の補足(復習): Ex02_2.zipのダウンロード

www.riken.jp/brict/Yoshizawa/Lectures/index.html



- ✓ firefoxを使ってEx02_2.zipを第二回演習のディレクトリーにダウンロード.

演習の補足(復習): Ex02_2.zipの展開

- ✓ ファイルブラウザを使う場合:
  1. ファイルブラウザーを立ち上げる.
  2. Ex02_2.zipをダウンロードしたディレクトリーを開く.
  3. Ex02_2.zipを右クリックしてメニューから展開を選ぶ.
- ✓ 端末を使う場合:
  1. 端末を立ち上げる.
  2. 端末で「cd ~/学籍番号_Ex02」を入力しエンターキーを押す.(lsでEx02_2.zipがある事を確認！)
  3. 端末で「unzip Ex02.zip」を入力しエンターキーを押す.

演習の補足(復習): プログラムの実行方法

全部自分で端末に打ち込む場合:

1. 端末を立ち上げる.
2. Ex02_2を展開したディレクトリーを開く.
   端末にて「cd ~/学籍番号_Ex02/Ex02_2」を入力しエンターキーを押す.
3. 端末にて
   「./実行ファイル名 入力画像ファイル名 出力画像ファイル名」を入力しエンターキーを押す.例えば
   「./ex02_0 lena.pgm test.pgm」や
   「./ex02_1 lena.ppm test.ppm」
4. 端末にて「display 出力画像ファイル名 &」で確認.例えば「display test.ppm &」

演習の補足(復習): その他

1. 画像の見方:端末で「display 画像ファイル名」を入力しエンターキーを押す.
2. 実行ファイルの実行の仕方:端末で「./実行ファイル名」を入力してエンターキーを押す.
   注意:「./」を付けるのを忘れずに！
3. C/C++でのコメントアウト「//」又は「/*」と「*/」で囲む.
4. プログラムのソースファイルの編集方法(emacs):
   1. ファイルブラウザーを使う場合: ファイルブラウザーで開きたいファイルを右クリックして、メニューからemacsで実行を選ぶ.
   2. 端末を使う場合: 端末にて「emacs ファイル名」を入力してエンターキーを押す.