情報デザイン専攻

画像情報処理論及び演習II

-動画像処理-
基礎、Video Stylization

第10回講義
水曜日１限
教室6218

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

今日の授業内容

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec23.pdf

1. 連番画像とビデオ.
2. Artistic Stylization ⇒ Video Stylization
3. 演習：DoG画像、DoGビデオ、Artistic Stylization

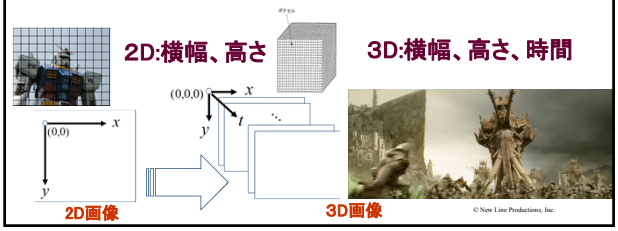
レポート05は今日〆切です！

重要：↑は次回レポートの内容なので頑張ってp(^ ^)q
＋今日作るプログラム(クラス)を次回以降の演習で使うので必ず来週までに作成してください！

レポート04の結果を取りに来てください！

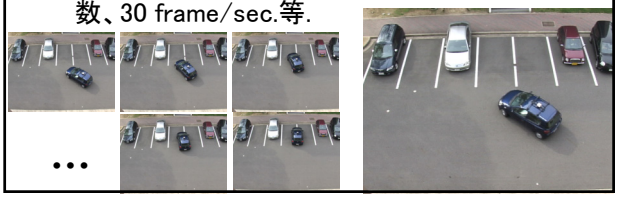
動画像の基礎

- ✓ 動画像フォーマット:
  - ASF(wmv等), AVI, MPEG (mpg,mp4等), DVD, RealVideo, DviX, Flash(flv), QuickTime, MP4,…
  - Animated Gif, multipage TIFF, …
- ✓ 理論/数学的には1次元増えただけ⇒3D画像.



動画像の基礎2

- ✓ 講義では複数の2D画像の組で3D画像を扱う.
  - 画素：ピクセル(2D)→ボクセル(3D).
  - サイズ:(sx,sy)→(sx,sy,st).
  - 輝度値：2次元配列→3次元配列.
  - ループ:　2重→3重.
  - フレームレート: 単位時間のフレーム(2D画像)数、30 frame/sec.等.

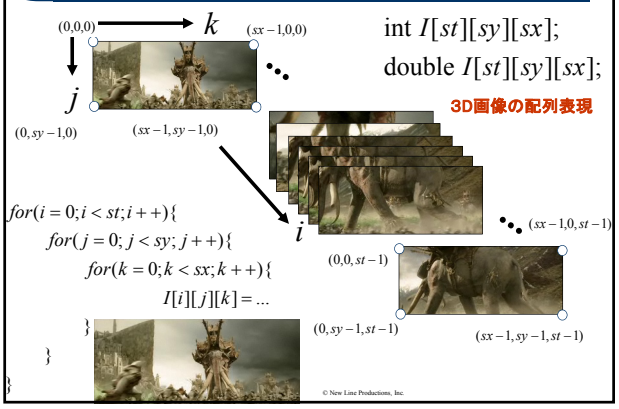


動画像の基礎3

- ✓ 複数2D画像ファイル⇔動画フォーマットの変換:
  - 符号化方式(ファイルフォーマット)を用いてデータのencode/decodeを行うコーデックが必要.
  - フリーのソフトを使うのが簡単で良い.
  - 例えばWinでは、AVIMaker(bmp→avi)やAviUtl(bmp⇔avi):
    http://www.vector.co.jp/soft/dl/win95/art/se121264.html
    http://spring-fragrance.mints.ne.jp/aviutl
  - http://www.vector.co.jpに色々な動画⇔画像ソフトがあるので、みんな独自のビデオを連番bmp画像にしてみましょう！
  - Linuxでは機能が多彩で難しい！画像・動画⇔動画: ffmpeg
  - 簡単！複数bmp⇔gifアニメ(Linux): convert
  - 動画へ「convert *.bmp 出力.gif」
  - 画像へ「convert 入力.gif 出力.bmp」

番号を揃えたい場合はCのprintfの表記と同じに
「convert 入力.gif 出力%0桁数d.bmp」とする. 例えば3桁なら
「convert 入力.gif 出力%03d.bmp」

動画像の配列表現



Shin Yoshizawa: shin@riken.jp

動画像の数式表現

輝度値の数式表現：高次元の高さ関数

$z = I(x, y, t)$ **又は** $z = I(\mathbf{x}), \quad \mathbf{x} = (x, y, t)$

カラー画像： $\mathbf{z} = \mathbf{I}(x, y, t) = (R(x, y, t), G(x, y, t), B(x, y, t))$

又は $\mathbf{z} = \mathbf{I}(\mathbf{x}) = (R(\mathbf{x}), G(\mathbf{x}), B(\mathbf{x})), \quad \mathbf{x} = (x, y, t)$

Shin Yoshizawa: shin@riken.jp

Image3Dクラスの使い方

✓ 使い方は今まで使ってきたSimpleImage.hのImageクラスとほぼ同じで、一次元増えただけ.

Image3D* 変数名 = new Image3D();

か

Image3D* 変数名 = new Image3D(サイズ);

例えば横500×縦256の画像が120枚あった場合に3D画像を

Image3D *AAA = new Image3D(500,256,120);とし

for(int i=0;i<AAA->st;i++)

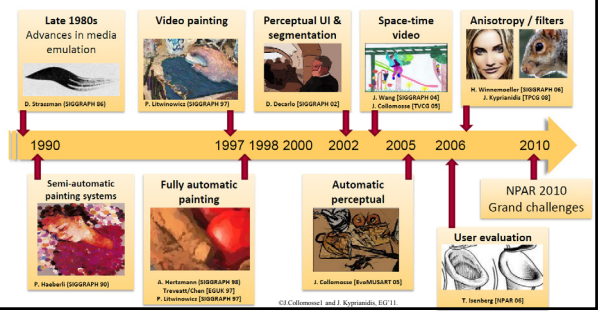
　for(int j=0;j<AAA->sy;j++)　使い終わったらdelete AAA;を忘れずに!

　　for(int k=0;k<AAA->sx;k++)AAA->img[i][j][k]で輝度値を参照する. カラーの場合は三つのImage3D

Shin Yoshizawa: shin@riken.jp

復習：Artistic Stylization

✓ アーティストの様式を疑似的に再現して実画像を生成・編集する事: NPR/計算Photographyの分野.



Shin Yoshizawa: shin@riken.jp

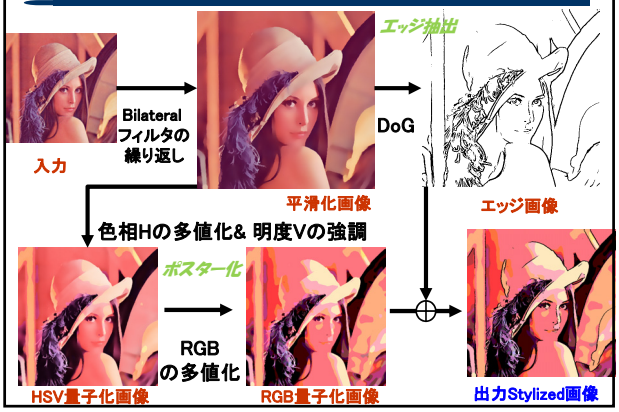
Artistic Video Stylization

✓ 2Dの基本フレームワークを3D化してみよう！

エッジ保存平滑化→エッジ抽出→ポスター化(多値化、量子化)→合成.

Shin Yoshizawa: shin@riken.jp

2Dの基本フレームワーク



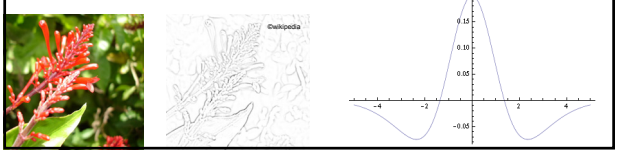
Shin Yoshizawa: shin@riken.jp

復習：DoG

✓ DoG: Difference of Gaussian.

$$g_\sigma(x, y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

$$DoG_{\sigma,K}(x, y) = g_\sigma(x, y) - g_{K\sigma}(x, y)$$



Shin Yoshizawa: shin@riken.jp

DoG2

✓ DoGと入力画像の畳み込みが負の領域=エッジ:

Shin Yoshizawa: shin@riken.jp

復習：Bilateralフィルタとは？

Gaussian Filter ⇦ Input ⇨ Bilateral Filter

$$Z(\mathbf{x},\mathbf{y}) = g_\sigma(|\mathbf{x}-\mathbf{y}|)$$

$$g_a(r) = e^{-\frac{r^2}{a^2}}.$$

$$Z(\mathbf{x},\mathbf{y}) = g_h(|I(\mathbf{x})-I(\mathbf{y})|)\,g_\sigma(|\mathbf{x}-\mathbf{y}|)$$

Intensity (Tonal) Kernel

Spatial Kernel

Spatial-Tonal Normalized Convolution:

$$I^{\text{new}}(\mathbf{x}) = \int Z(\mathbf{x},\mathbf{y})I(\mathbf{y})d\mathbf{y} \Big/ \int Z(\mathbf{x},\mathbf{y})d\mathbf{y},$$

エッジ特徴を保存する！

Shin Yoshizawa: shin@riken.jp

Bilateralフィルタの繰り返し適用

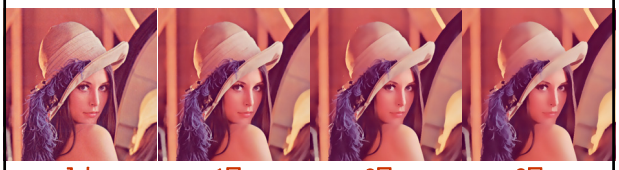
✓ エッジ保存平滑フィルタを繰り返し適用するとエッジに沿った領域が断片化される(領域抽出効果):

```
I^{n+1} = Filtering(I^n)
for(iteration){
  tmp = Filtering(I);
  I = tmp;
}
```

$$I^{\text{new}}(\mathbf{x}) = \int Z(\mathbf{x},\mathbf{y})I(\mathbf{y})d\mathbf{y} \Big/ \int Z(\mathbf{x},\mathbf{y})d\mathbf{y},$$

$$Z(\mathbf{x},\mathbf{y}) = g_h(|I(\mathbf{x})-I(\mathbf{y})|)\,g_\sigma(|\mathbf{x}-\mathbf{y}|)$$

$$g_a(r) = e^{-\frac{r^2}{a^2}}.$$



入力 1回 2回 3回

$\sigma = 25.0, \quad h = 0.1$ ×輝度値の標準偏差

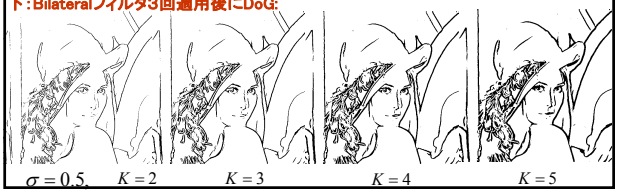
Shin Yoshizawa: shin@riken.jp

DoG+Bilateralフィルタ

✓ Bilateralフィルタを繰り返し適用後にDoGを適用:

上：入力画像にDoG:

下：Bilateralフィルタ3回適用後にDoG:



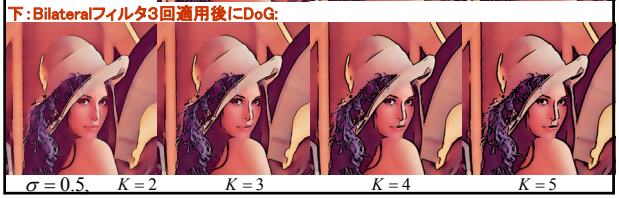
Shin Yoshizawa: shin@riken.jp

DoG+Bilateralフィルタ

✓ Bilateralフィルタ後の画像と合成すると…

上：入力画像にDoG:

下：Bilateralフィルタ3回適用後にDoG:



Shin Yoshizawa: shin@riken.jp

ポスター化

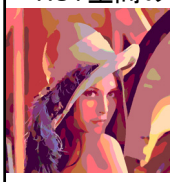
✓ 多値化で量子化する事でポスター化:
- RGB毎に多値化すると色が混ざる.
- HSV空間の色相(H)で多値化し明度(V)を強調.

Bilateralフィルタ3回適用後　BGB毎に4段階の値へ量子化

Shin Yoshizawa: shin@riken.jp

ポスター化2

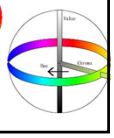
✓ HSV空間の色相(H)で多値化し明度(V)を強調.



✓ 色相(Hue): 色の様相の相違: 光の波長の様相.

✓ 彩度(Saturation/Chroma): 鮮やかさ.

✓ 明度(brightness/value/intensity): 明るさ.

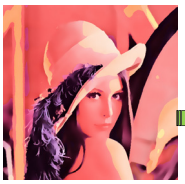


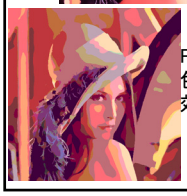
色相を16段階の値へ量子化+明度を強調.

Shin Yoshizawa: shin@riken.jp

ポスター化3

✓ HSV空間の色相(H)で多値化し明度(V)を強調.





RGB毎の混色で鏡面的効果を演出.

色相を16段階の値へ量子化+明度を強調
+RGB毎に4段階に多値化.

Shin Yoshizawa: shin@riken.jp

今週はVideoへのDoG拡張

Shin Yoshizawa: shin@riken.jp

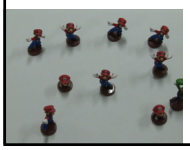
DoGの3D拡張

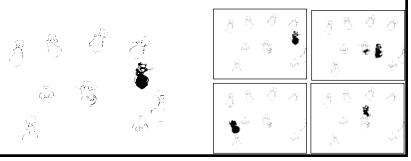
✓ そのままの拡張は時空間エッジになるので Artistic Stylizationでは工夫が必要:

- 注意点:時間方向のパラメータhは空間と分けなければダメ、時間方向の畳み込み半径も同様.

$$g_{\sigma,h}(x,y,t)=\frac{1}{2\pi\sigma^2}\frac{1}{\sqrt{2\pi}h}\exp\left(-\frac{x^2+y^2}{2\sigma^2}-\frac{t^2}{2h^2}\right)$$

$$DoG_{\sigma,K,h}(x,y,t)=g_{\sigma,h}(x,y,t)-g_{K\sigma,Kh}(x,y,t)$$



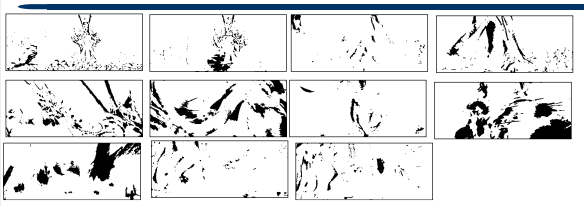


Shin Yoshizawa: shin@riken.jp

DoGの3D拡張2

Shin Yoshizawa: shin@riken.jp

DoGの3D拡張3



ストーリー展開の描写はOKだが単純に重ねるとあまり良くない.

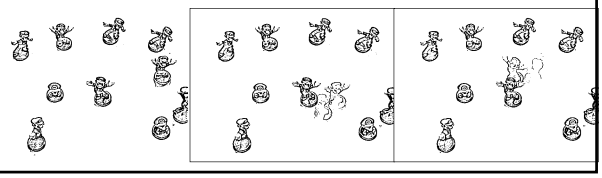
Shin Yoshizawa: shin@riken.jp

DoGの3D拡張4

- ✓ 2D空間DoGを時間方向に平滑化し、残像効果:
  - 注意点:レポートでは講義で紹介した時間方向の拡張の仕方以外でもデザインしてOK.

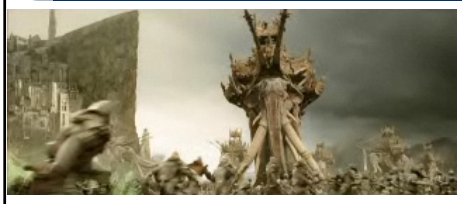
$$g_\sigma(x,y)=\frac{1}{2\pi\sigma^2}\exp(-\frac{x^2+y^2}{2\sigma^2}) \qquad g_h(t)=\frac{1}{\sqrt{2\pi}h}\exp\left(-\frac{t^2}{2h^2}\right)$$

$$DoG_{\sigma,K,h}(x,y,t)=g_h(t)(g_\sigma(x,y)-g_{K\sigma}(x,y))$$



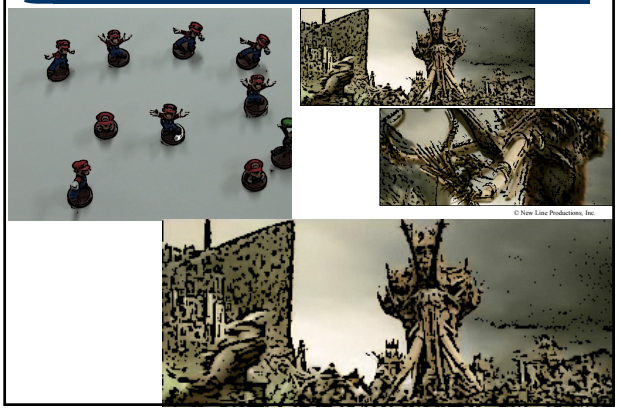
Shin Yoshizawa: shin@riken.jp

DoGの3D拡張5



Shin Yoshizawa: shin@riken.jp

DoGの3D拡張6



Shin Yoshizawa: shin@riken.jp

DoGの3D拡張7

- ✓ レポートでは時間方向拡張の仕方をデザインしてOK.
  ただし狙ったデザインの目的と使った数式を明記する事.
- ✓ パラメータの調節が必要.

Shin Yoshizawa: shin@riken.jp

演習:DoG画像、DoGビデオ

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec23.pdf
www.riken.jp/brict/Yoshizawa/Lectures/Ex15.zip

1. Ex15内のプログラムを動かしてみる.
2. DoGビデオプログラムの作成.

Shin Yoshizawa: shin@riken.jp

演習:Ex23-1

- ✓ Ex15.zip内でmakeでコンパイルし、testVideoIO.cxx、DoGEdge.cxx、DoGEdge2.cxx、Style.cxxを動かしてみる.
- ✓ 連番画像の入出力: VideoIO.h
  void OpenVideo(char *入力フォルダー名, Image3D *R, Image3D *G, Image3D *B, int *sx, int *sy, int *st);
  void SaveVideo(char *出力フォルダー名, char * 出力ファイル名, Image3D *R, Image3D *G, Image3D *B);
- ✓ DoGEdge.cxx: DoGによるエッジ画像の作成: 引数3.

./DoGEdge 畳み込み半径(int) DoG標準偏差(double) DoGバンド幅(double)

「./DoGEdge lena.bmp ex22_1_1.bmp 10 0.5 2」、
「./DoGEdge lena.bmp ex22_1_2.bmp 10 0.5 3」、
「./DoGEdge lena.bmp ex22_1_3.bmp 10 0.5 4」、
「./DoGEdge lena.bmp ex22_1_4.bmp 10 0.5 5」を実行して!

Shin Yoshizawa: shin@riken.jp

演習:Ex23-1

- ✓ DoGEdge2.cxx: DoGエッジと元画像の合成(引数3, DoGEdgeと同じ): DoGEdgeと同じパラメータで出力ファイル名を変えて実行してみましょう！
- ✓ Style.cxx: Artistic Stylization画像の作成(引数11).

./Style 畳み込み半径(int) DoG標準偏差(double) DoGバンド幅(double) Bilateralフィルタ空間標準偏差(double) Bilateralフィルタ輝度標準偏差(double) Bilateralフィルタ繰り返し回数(int) HSV量子化数(int) HSV量子化V強調パラメータ(double) RGB量子化数(int)

「./Style lena.bmp ex22_st_1.bmp 0 0.5 3.0 25.0 0.1 3 16 0.7 4」と「./Style lena.bmp ex22_st_1.bmp 10 0.5 5.0 25.0 0.1 3 16 0.7 4」で実行してみましょう！

- ✓ 自分の画像でDoGEdge.cxxとStyle.cxxをパラメータを調節してスタイリッシュな画像にしてみてください.

Shin Yoshizawa: shin@riken.jp

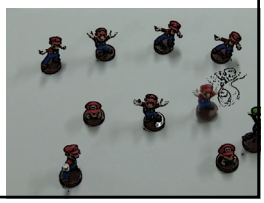
演習:Ex23-2

- ✓ DoGVideoEdge.cxxとDoGVideoEdge2.cxxを編集し、連番画像のDoGエッジ動画を作成するプログラムを完成せよ. ヒント:ファイル内のコメントとDoGEdge.cxxをよく見てみてください.

$$g_\sigma(x,y)=\frac{1}{2\pi\sigma^2}\exp(-\frac{x^2+y^2}{2\sigma^2}) \qquad g_h(t)=\frac{1}{\sqrt{2\pi}h}\exp\left(-\frac{t^2}{2h^2}\right)$$

$$DoG_{\sigma,K,h}(x,y,t)=g_h(t)(g_\sigma(x,y)-g_{K\sigma}(x,y))$$

- ✓ ↑と同じでもOK、自分独自の拡張でもOK、ただし単純拡張はダメ.
- ✓ ↑は⇒の様に残像だけなので、評価時刻でのDoGエッジも出るようにデザインすると高得点！



Shin Yoshizawa: shin@riken.jp

来週の予定

内容(10－13):動画像処理基礎、スタイル化合成等.

1回	画像フォーマット
2回	
3回	周波数分解
4回	
5回	
6回	フィルタ処理・エッジ強調
7回	
8回	
9回	計算Photography・Artistic Stylization
10回	
11回	動画像処理
12回	
13回	
14回	エッジ・形状・特徴抽出とパターン認識の基礎＋補講
15回	

Shin Yoshizawa: shin@riken.jp

参考資料:
Image3Dクラス

Shin Yoshizawa: shin@riken.jp

3D画像クラスの作成

- ✓ 3D画像クラス: Image3DクラスをSimpleImage3D.hというヘッダーファイル名で作ってみる.
  www.riken.jp/brict/Yoshizawa/Lectures/Ex14.zip
- ✓ 必要なクラスのメンバー/メソッド:
  - 画像サイズ(int)で三つsx,sy,st.
  - 輝度値を格納するためのdoubleの3重ポインター.
  - コンストラクター二つ:
    - 引数無: サイズにゼロ、輝度値のポインターにNULLを代入する.
    - 引数画像サイズ: 輝度値の3重ポインターのメモリを確保して3次元配列にする.
  - デストラクター:クラスがdeleteしたとき輝度値の3次元配列をdeleteする.

Shin Yoshizawa: shin@riken.jp

C++クラスの基礎

```
class クラス名{   /* 設計図の様なものでクラス=新しい型 */
  public:   /* パブリックの場合は、クラスの外から参照可能 */
  メンバー変数   /* クラスが持っている変数、構造体、クラス内クラス */
  クラス名(){   /* コンストラクター:newされたときに呼ばれる. */
  }
  クラス名(引数){} /* コンストラクターは複数あってよい */

  ~クラス名(){   /* デストラクター:delete されたときに呼ばれる. */
  }
  戻り値 メソッド名(引数){} /* メソッドを作れる= */
  private: /* プライベートの場合は、クラスの外から参照不可 */
};
```

多重ポインターから多次元配列を作る方法

✓ 1重ポインターから1次元配列を作る方法:

```
double *A = new double[N];
```

これで、A[0], A[1], …A[N-1]まで配列として使える.

- 使い終わったらメモリの開放が必要:`delete [] AAA;`

✓ 2重ポインターから2次元配列を作る方法:

```
double **A = new double *[N];
for(int i=0;i<N;i++)A[i] = new double[M];
```

これで、A[0][0], A[0][1], …A[0][M-1], A[1][0], A[1][1],…A[N-1][M-1]まで配列として使える.

- 使い終わったらメモリの開放が必要:

```
for(int i=0;i<N;i++) delete [] A[i];
delete [] A;
```


多重ポインターから多次元配列を作る方法2

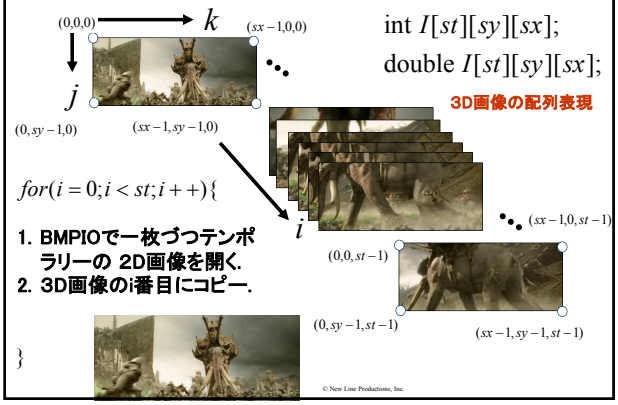
✓ 3重ポインターから3次元配列を作る方法:

```
double ***A = new double **[st];
for(int i=0;i<st;i++){
    A[i] = new double *[sy];
    for(int j=0;j<sy;j++)A[i][j] = new double[sx];
}
```

これで、A[0][0][0], A[0][0][1], …A[0][0][sx-1], A[0][1][0], A[0][1][1], …A[0][sy-1][sx-1], A[1][0][0], A[1][0][1],…A[st-1][sy-1][sx-1]まで配列として使える. 同様にメモリの開放は以下:

```
for(int i=0;i<st;i++){
    for(int j=0;j<sy;j++) delete [] A[i][j];
    delete [] A[i];
}
delete [] A;
```


連番画像の入出力へ向けて



連番画像名の取得方法

✓ Ex14.zip内のImageSetIO.cxxを開いてください.入力としてフォルダー名を与えて、その中のBMPファイルをファイル名順にソートしたファイル名のリストを得るプログラムです.

✓ 今回の演習でやる方法は、

ステップ1:Linux/UnixコマンドのlsとgrepをC/C++からシステムコール関数system()を使って、与えられたフォルダー名内のBMP画像ファイル名(複数)をテンポラリーのファイル(tmp_img_file_names.txt)に書き出す.

- system()はstdlib.hが必要.
- system(char*)で引数に書いたLinuxコマンドを実行出来る. 例:system("ls");

連番画像名の取得方法2

✓ 今回は以下のコマンドを用いる:

"ls 入力フォルダー名 | grep .bmp > 出力ファイル名"

ここで|と>はそれぞれ、パイプとリダイレクトと呼ばれてコマンドの結合とファイルへの出力を行える:

- 「ls AAA」 AAA内のファイル名・フォルダー名を出力する.
- 「grep AAA BBB」 BBBの中からAAAがある行を抜き出す.
- 「AAA | BBB」 AAAの結果をBBBに渡す.
- 「AAA > BBB」 AAAの結果をBBBに書き出す.
- sprintf(格納先,printfの表記,変数)でコマンド内にメインの引数やテンポラリーファイル名をプリント.

与えられたフォルダー名内のlsの結果から.bmpが付いているファイル名だけ抽出して出力ファイルに書き出すコマンド.

連番画像名の取得方法3

✓ ステップ2:テンポラリーのファイル(tmp_img_file_names.txt)を開いて、一行づつfscanf()で呼び込み、vector<char *>へ格納する:

- FILE *fp = fopen(ファイル名,"r");で開いたファイルポインターfpを使ってfscanf(fp,"%s",格納先)の戻り値がEOFでない間、繰り返しスキャンする.
- vectorを使うには#include<vector>が必要.
- vector<char *>へ代入するためにchar *をnewしてfscanf()の結果をコピーする.
- .push_back()メソッドを使ってvectorへ格納する.

格納後はvectorなので配列の様に使える. 例えば、vector<char *> AAA;ならAAA[0]に最初のファイル名がchar *で入っており、以下AAA[1], AAA[2]と使える. サイズ(push_backした回数=ファイル名の数)は AAA.size()で得られる.

Shin Yoshizawa: shin@riken.jp

連番画像名の取得方法4

- ✓ **ステップ3**：**std::sort**を使ってvector<char *>に格納したファイル名をソートする．例えばvector <char *> AAA;なら**std::sort(AAA.begin(),AAA.end());**でソートされる．
  - std::sortは#include<algorithm>が必要．
- ✓ **ステップ4**：ソート後は、vector<char *>を配列の様に使いファイル名の操作を行い、実際の処理をする．
  - ImageSetIO.cxxは連番名の取得だけなので、実際の処理は無いが、演習ではVideoIO.cxxでソート後のファイル名を順番に開いて3D画像クラスに格納する．BMPIO.hを使って2D毎に入出力をファイル名の数だけ行う．
- ✓ **ステップ5**：new したchar *のメモリを解放する．例えば、

for(i=0;i<AAA.size();i++)delete [] AAA[i];

Shin Yoshizawa: shin@riken.jp

演習: 連番画像の入出力

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec23.pdf
www.riken.jp/brict/Yoshizawa/Lectures/Ex14.zip

1. Lec23-1: 3D画像クラスをSimpleImage3D.hとして作成せよ．
2. Lec23-2: 連番画像の入出力を行うプログラムVideoIO.cxxをコメントを読みながら作成せよ．LV3_1.zipとLV3_5.zipを展開して入力フォルダーとして実行してみよ．

Lec23-3: ↑の1,2を使って、連番の各画像にBilateralフィルタ(Lec20-2)を計算して結果を保存するプログラムを作成してみましょう．