情報デザイン専攻

画像情報処理論及び演習II

-計算Photography3-

Video Stylization

第11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

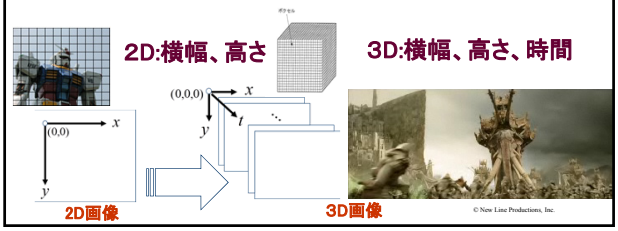
www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec23.pdf

1. 動画像の基礎
2. Video Stylization

動画像の基礎

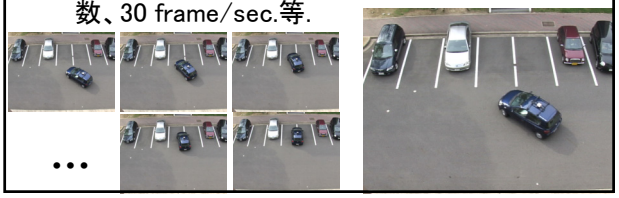
✓ 動画像フォーマット:
- ASF(wmv等), AVI, MPEG (mpg,mp4等), DVD, RealVideo, DviX, Flash(flv), QuickTime, MP4,…
- Animated Gif, multipage TIFF, …

✓ 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」

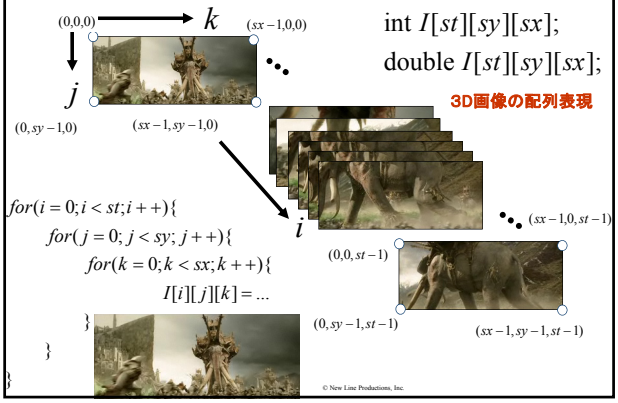
演習: gif anime

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

1. 上記URLからLV3_1.zip及びLV3_5.zipをダウンロード.
2. 右クリックで「展開」を選び(例えばデスクトップに)圧縮ファイルを展開.
3. 端末を開き、「cd」で展開したフォルダーへ移動.
4. 端末にて「convert *.bmp output.gif」でgif anime へ変換.
5. Webブラウザ(IE)にoutput.gifをドラッグ &ドロップ.

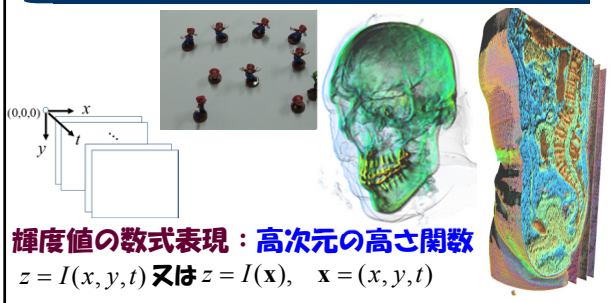
Shin Yoshizawa: shin@riken.jp

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

例えば、Image3Dクラス

✓ 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(); 又は、
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++)
　　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を使う.

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 */
};
```

Shin Yoshizawa: shin@riken.jp

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;
```

Shin Yoshizawa: shin@riken.jp

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];
}
```

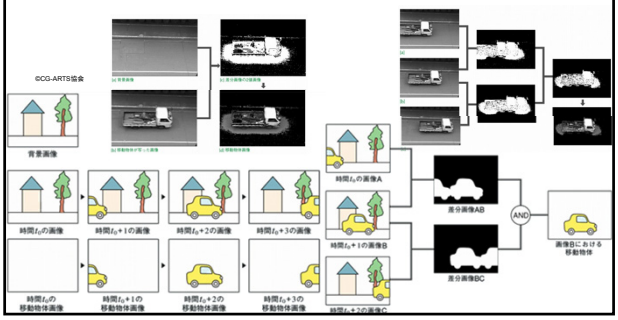
これで、A[0][0][0], A[0][0][1], …A[0][0][sx−1], A[0][1][0], A[0][1][1], …A[0][sy−1][sx−1], A[1][0][0], A[1][0][1],…A[st−1][sy−1][sx−1]まで配列として使える. 同様にメモリの開放は以下:

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

Shin Yoshizawa: shin@riken.jp

動画のパターン認識

- ✓ 基本は静止画のパターン認識法を高次元として適用する:
  - 背景差分、オプティカルフロー、パーティクルフィルタ、確率論等.
- ✓ 背景・フレーム間差分：時間微分の差分近似.

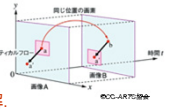


Shin Yoshizawa: shin@riken.jp

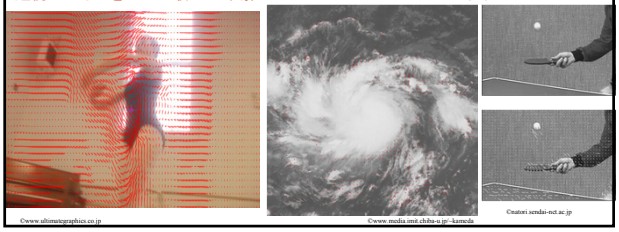
動画のパターン認識2

- ✓ オプティカルフロー: 移動物体の運動解析.
  - ブロックマッチング法：テンプレートマッチング.
  - 勾配法：

$$\frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v+\frac{\partial I}{\partial t}=0,$$



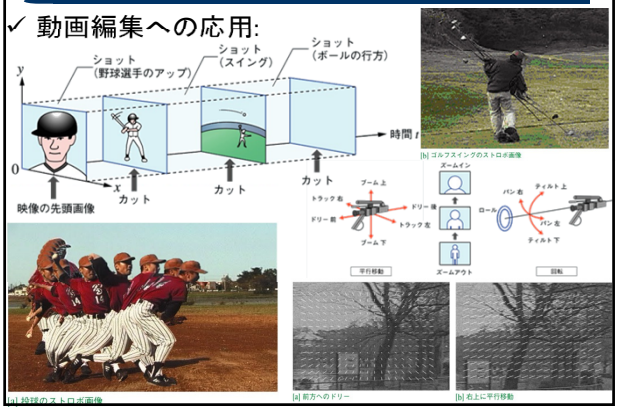
近傍からも式を立てて最小二乗解.



Shin Yoshizawa: shin@riken.jp

動画のパターン認識3

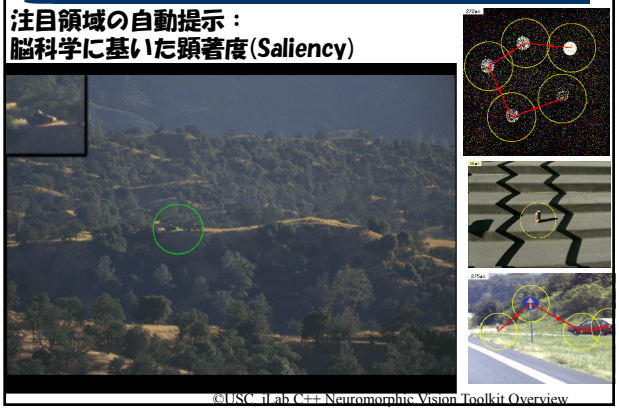
- ✓ 動画編集への応用:



Shin Yoshizawa: shin@riken.jp

応用例：コンピュータ・ビジョン

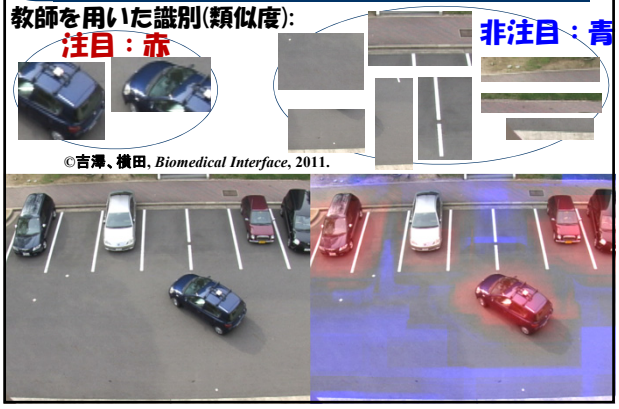
注目領域の自動提示：
脳科学に基いた顕著度(Saliency)



Shin Yoshizawa: shin@riken.jp

応用例：パターン認識

教師を用いた識別(類似度):

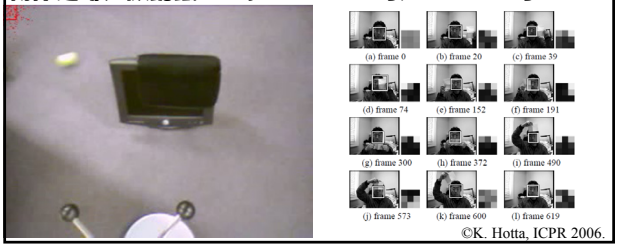


Shin Yoshizawa: shin@riken.jp

応用例：パターン認識

Google等の画像検索：リトリーバル

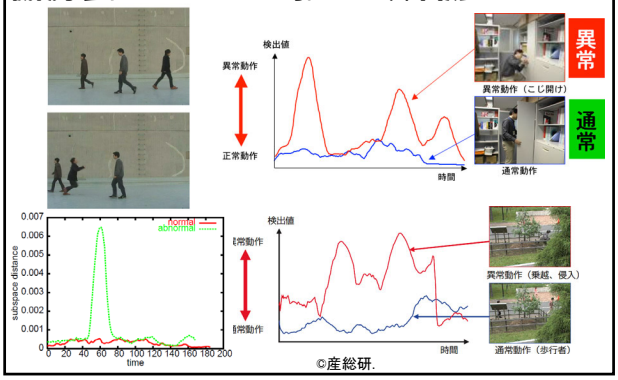
物体追跡、顔認識：Object Tracking, Face Recognition



Shin Yoshizawa: shin@riken.jp

応用例：パターン認識

機械学習(Machine Learning)による異常検出：



©産総研.

Shin Yoshizawa: shin@riken.jp

応用例：Content-Aware Resizing

✓ Seam Carving: ビデオへも拡張＆マスクと組み合わせてオブジェクトの削除も.

Shin Yoshizawa: shin@riken.jp

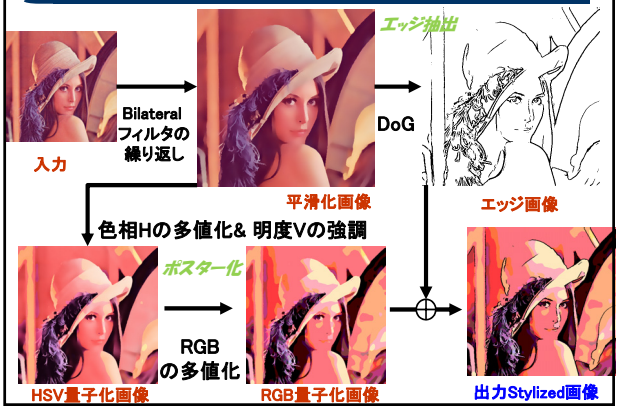
Artistic Video Stylization

✓ 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.

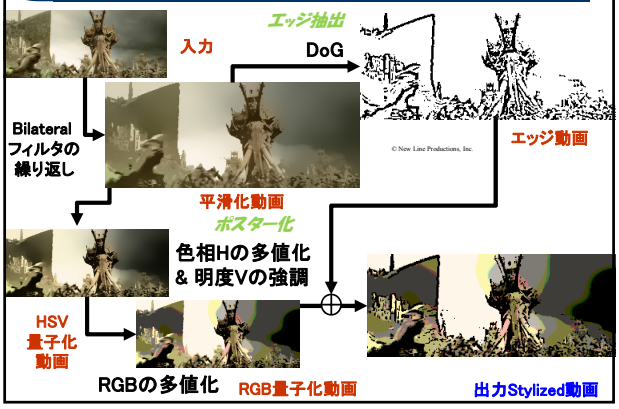
Shin Yoshizawa: shin@riken.jp

2Dの基本フレームワーク



Shin Yoshizawa: shin@riken.jp

スタイル化ビデオ



Shin Yoshizawa: shin@riken.jp

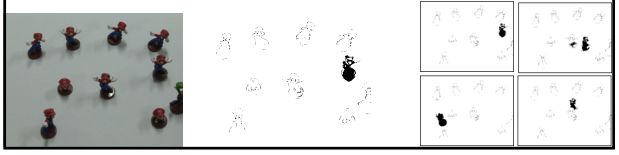
DoGの3D拡張

✓ 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.

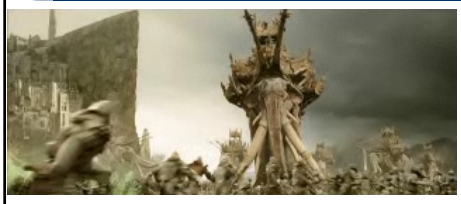
$$g_{\sigma,h}(x,y,t)=\frac{1}{2\pi\sigma^2}\frac{1}{\sqrt{2\pi}h}\exp(-\frac{x^2+y^2}{2\sigma^2}-\frac{t^2}{2h^2})$$

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



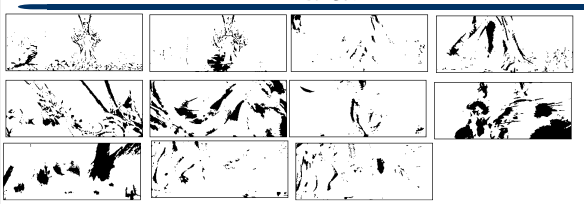
Shin Yoshizawa: shin@riken.jp

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.

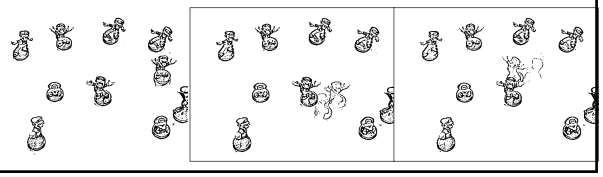
Shin Yoshizawa: shin@riken.jp

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: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.

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

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

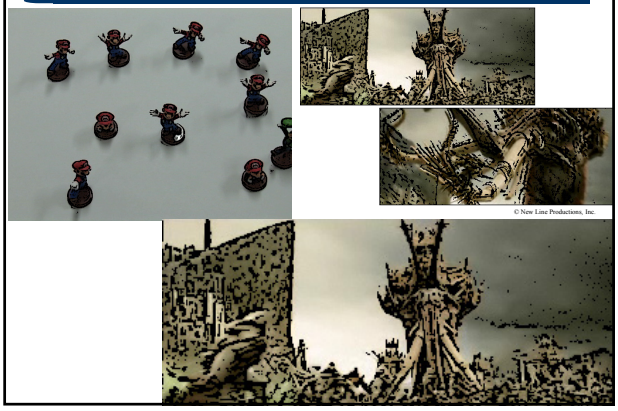


Shin Yoshizawa: shin@riken.jp

DoGの3D拡張5

Shin Yoshizawa: shin@riken.jp

DoGの3D拡張6



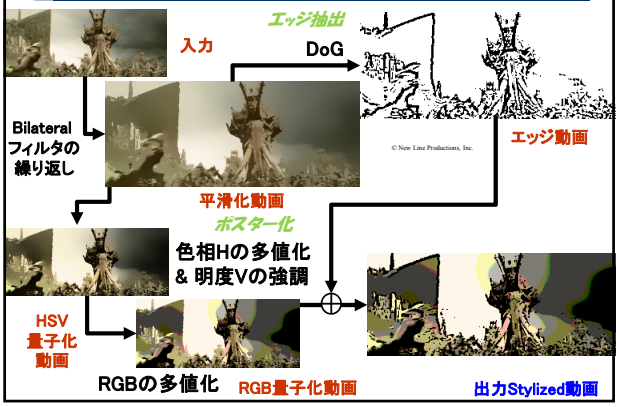
Shin Yoshizawa: shin@riken.jp

DoGの3D拡張7

✓ 時間方向拡張の仕方をデザインしてもOK.
✓ パラメータの調節が必要.

Shin Yoshizawa: shin@riken.jp

スタイル化ビデオ



Shin Yoshizawa: shin@riken.jp

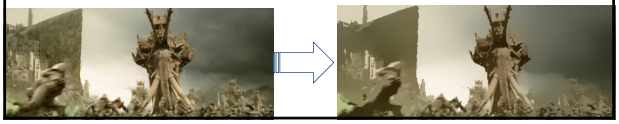
時空間Bilateralフィルタ

✓ 単純に時間方向のガウス関数を追加するだけでOK.

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

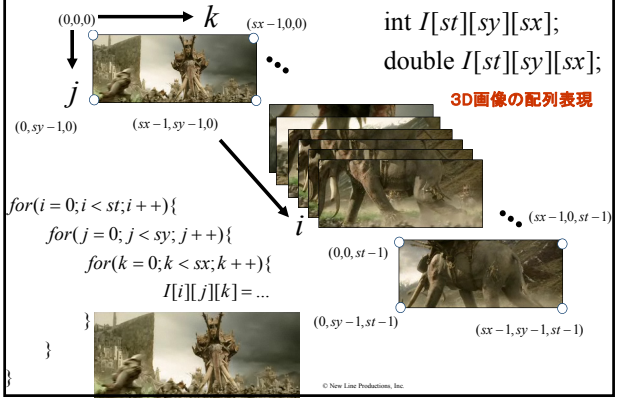
$$g_a(r) = e^{-\frac{r^2}{a^2}}. \qquad Z(\mathbf{x},\mathbf{y}) = g_h(|I(\mathbf{x}) - I(\mathbf{y})|)g_\sigma(|\mathbf{x}-\mathbf{y}|)g_\gamma(|\alpha-\beta|)$$

Intensity Kernel　Spatial Kernel　Temporal Kernel



Shin Yoshizawa: shin@riken.jp

復習：動画像の配列表現



Shin Yoshizawa: shin@riken.jp

量子化の3D拡張

✓ そのままの拡張は時間変化に弱いので、時間方向の半径を考えて、その半径内(部分画像毎)に量子化を実行する:

✓ 例えばHSV量子化では…

```
for(i = 0;i < st;i++){
  for(j = 0;j < sy;j++){
    for(k = 0;k < sx;k++){
      I[i][j][k] = ...
    }
  }
}
```

単純な3D化

```
for(i = 0;i < st;i++){
  for(t = i − r;t <= i + r;t++){
    for(j = 0;j < sy;j++){
      for(k = 0;k < sx;k++){
        I[t][j][k] = ...
      }
    }
  }
}
```

RGB⇒HSV

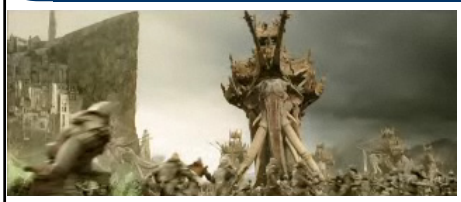
Hの多値化＋Vの強調

```
for(j = 0;j < sy;j++){
  for(k = 0;k < sx;k++){
    I[i][j][k] = ...
  }
}
```

HSV⇒RGB

Shin Yoshizawa: shin@riken.jp

量子化の3D拡張2

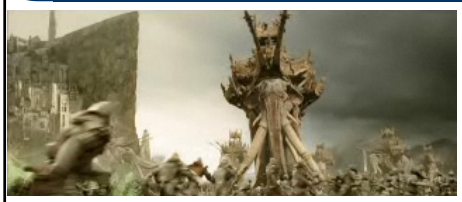


入力：256³色

色相Hは16段階、明度V強調0.5 時間半径16

Shin Yoshizawa: shin@riken.jp

量子化の3D拡張3

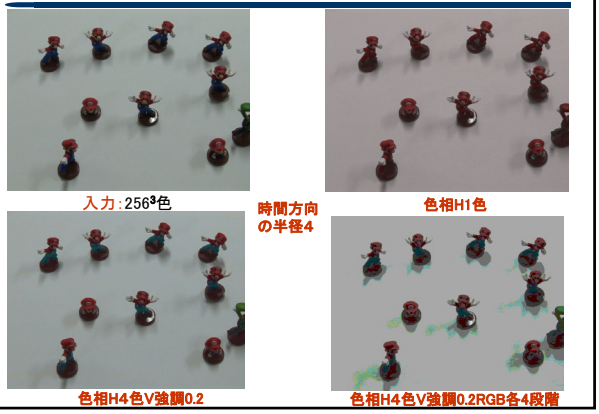


入力：256³色

色相Hは16段階、明度V強調0.5 RGB各4段階 時間半径16

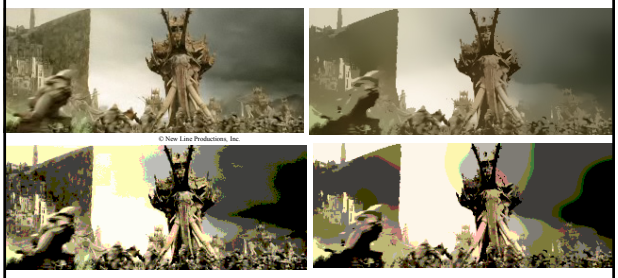
Shin Yoshizawa: shin@riken.jp

量子化の3D拡張4



Shin Yoshizawa: shin@riken.jp

量子化の3D拡張5

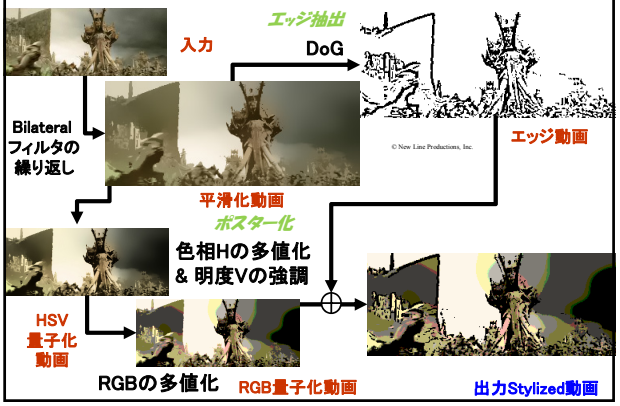


色相Hは16段階、明度V
強調0.5
RGB各4段階
時間半径16

**Bilateralフィルタ
3回適用後を入力**

Shin Yoshizawa: shin@riken.jp

復習：スタイル化ビデオ



Shin Yoshizawa: shin@riken.jp

演習:量子化ビデオ、スタイルビデオ

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. VideoStyle.cxxは完成しているので、Ex15.zipをダウンロードし、圧縮ファイルを展開.
2. 端末にて「make」でコンパイル.
3. 出力用フォルダーを作成し、英語(半角英数のみ)で名前を付ける. 例えばEx15の中に「Test」という名前のフォルダーを作成.
4. 引数14で実行. 引数の意味は次のスライド.

✓ 実行例：注意：LV3_5がEx15と階層構造で同じ場所にあるとき. Ex15の中にある場合は ↓は「./LV3_5」.

./VideoStyle ../LV3_5 ./Test output 10 0.5 3.0 25.0 0.1 3 1 1.0 16 0.4 4

Shin Yoshizawa: shin@riken.jp

演習: VideoStyleの14の引数

1. 入力フォルダー名
2. 出力フォルダー名
3. 出力ファイル名(.bmpなし)
4. 畳み込みの半径(int)
5. DoGの半径(double)
6. DoGのバンド幅(double)
7. Bilateralフィルタの空間標準偏差(double)
8. Bilateralフィルタの輝度値標準偏差(double)
9. Bilateralフィルタの繰り返し回数(int)
10. 時間方向畳み込み半径(int)
11. Bilateral&DoGの時間標準偏差(double)
12. HSVのHを多値化する数(int)
13. HSV量子化のV強調パラメータ(double)
14. RGBを多値化する数(int)

Shin Yoshizawa: shin@riken.jp

おわりに、

みなさん良く頑張りましたd(>_・)

今日で本講義は終わりです.

みんな最後まで来てくれてありがとー
o(≧∇≦)o

またいつかお会いしましょう！
ヾ(^-^)ゞ