Moodleの
出席確認を
提出しておいて
下さい。

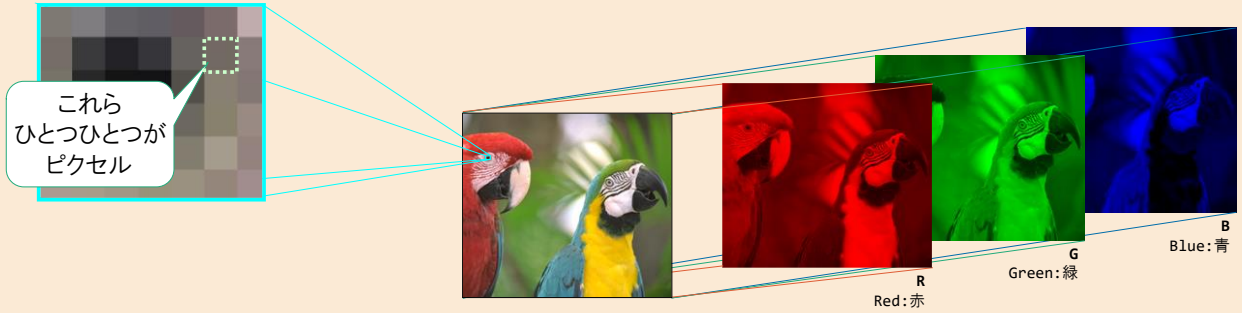
VisualStudio2019(等)で、
C言語+OpenCV のコーディングができる状態に
準備していてください。

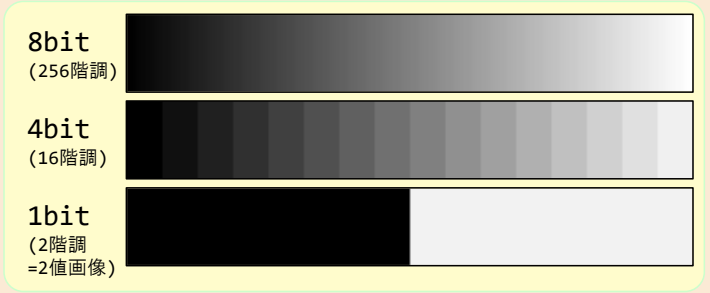
画像処理 (4J)

第14回

第6回のまとめ

- ラスタ画像とベクタ画像 ・・・ この授業では、ピクセル情報の集合であるラスタ画像を扱う
- 解像度 ・・・ 画像の大きさ(細かさ)
- ピクセル(画素) ・・・ ラスタ画像を構成する1つの点
- チャンネル ・・・ 1ピクセルをいくつの値で表現するか (例:RGBの3ch)
- 階調数 ・・・ 濃度を何段階で表現するか (例:8bit(=256段階))



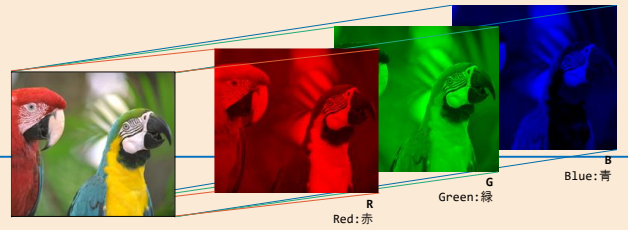


デジタル写真 = 有限の解像度で空間的にサンプリング(標本化)し、
有限の階調値で明るさを表現(量子化) したもの ...と捉えることができる。

※音声信号のデジタル化と対応させると、サンプリング周波数が解像度に、量子化bit数が階調数に、チャンネル数はそのまま対応する

第7回のまとめ

グレイスケール画像



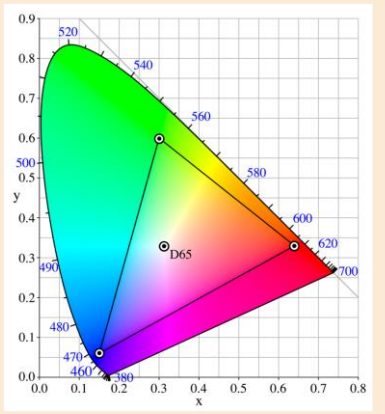
RGBカラー画像

●グレイスケール画像とカラー画像

- ➢グレイスケール画像は1つの(x,y)座標点に1つの濃度値 g(x,y)
- ➢RGBカラー画像は、1つの座標点に、3つの濃度値

●RGBカラー画像

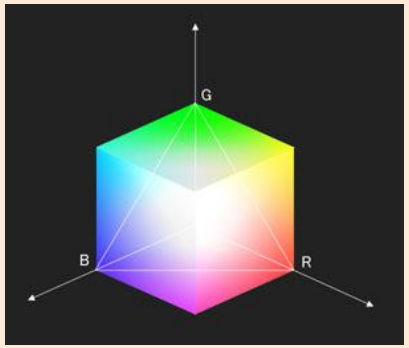
- ➢RGB値が同じでも、同じ色が表示されるとは限らない
- ➢sRGBに準拠させれば、一貫した色表現が可能。
  (ただし表現できる色域が狭い)

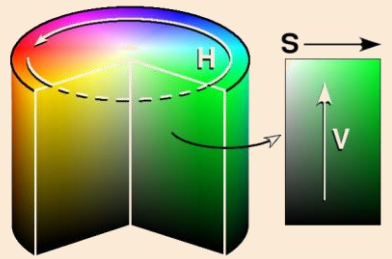


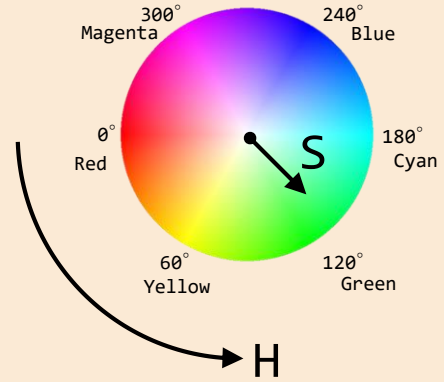
(CIE1931色空間のxy色度図上に示した)
sRGBの色域

●色空間： RGBとHSV

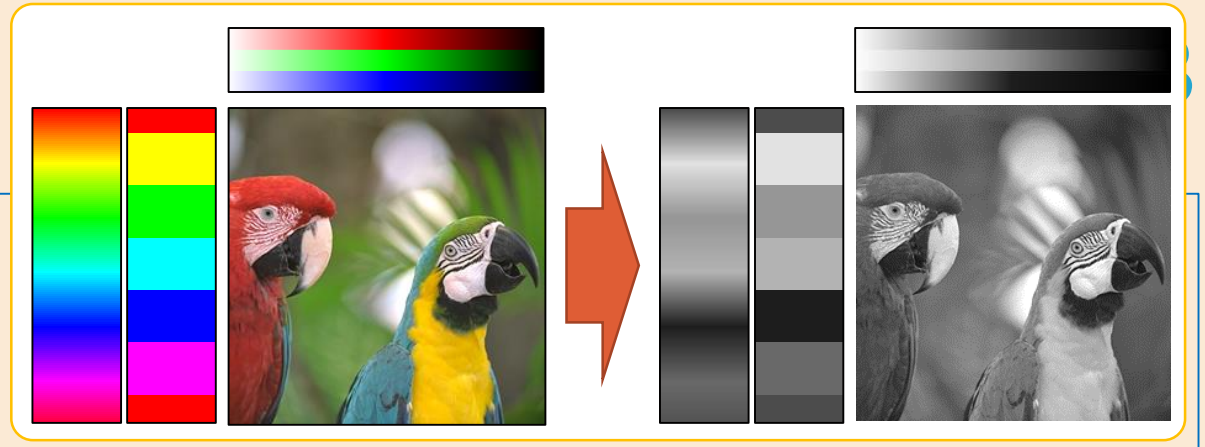
- ➢相互に変換可能
- ➢他にも様々な表色系がある







第8回まとめ



●グレイスケール化

➢**NTSC加重平均法**がよく使われる

$$Y = (0.298912 \times R + 0.586611 \times G + 0.114478 \times B)$$

●二値化

➢**閾値**を堺に、{0,1} の二値の画像に変換

➢閾値は任意に決められるが、画像統計量から閾値を自動決定する方法として**大津の方法**(判別分析法)が有名。

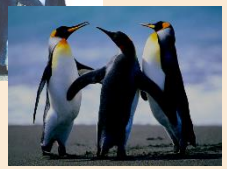
第9回まとめ

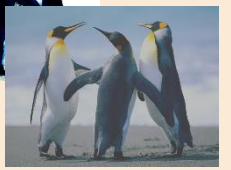
原画像 

階調反転

●濃度変換 ・・・ 濃度値を一定の方法で新しい濃度値に変換

- 線形変換 (Linear Stretch)　　$output = input \times a + b$
- ガンマ変換 (Gmma Stretch)　　$output = 255 \times \left(\frac{input}{255}\right)^{1/\gamma}$


輝度


コントラスト


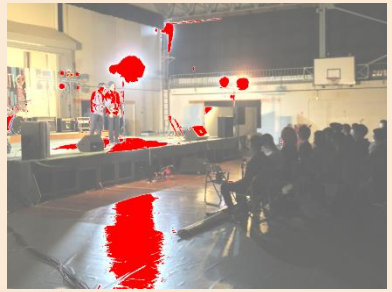
- 輝度調整、コントラスト調整、階調反転などに利用可能

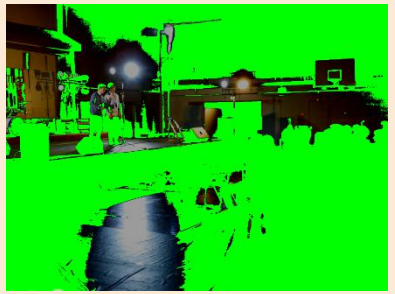


ガンマ

●濃度変換に伴う画像の劣化

- 白飛び　　・・・ 変換後に最大値以上になった場合に、最大値にクリップされる
- 黒つぶれ　　・・・ 変換後に最小値以下になった場合に、最小値にクリップされる
- 階調飛び(トーンジャンプ)　　・・・ 中間値の階調が失われ、濃度値が不連続に変化



白飛び



黒つぶれ

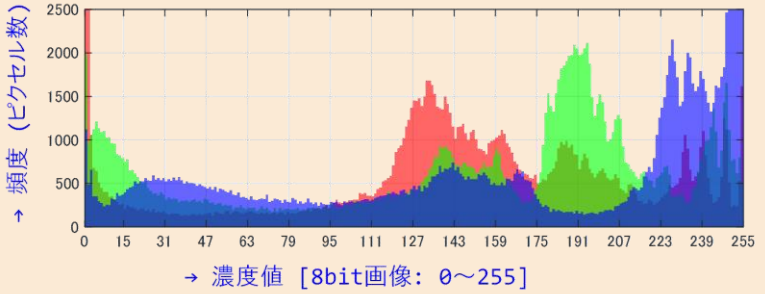


階調飛び

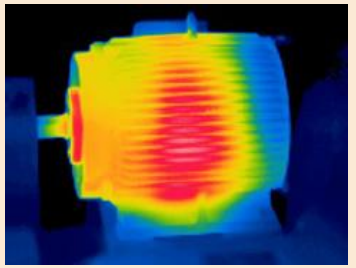
第10回まとめ

●ヒストグラム

- ➢濃度値の頻度(各濃度値が画像中にいくつあるか)を示したもの
- ➢ヒストグラムの形状から、画像の性質がある程度わかる

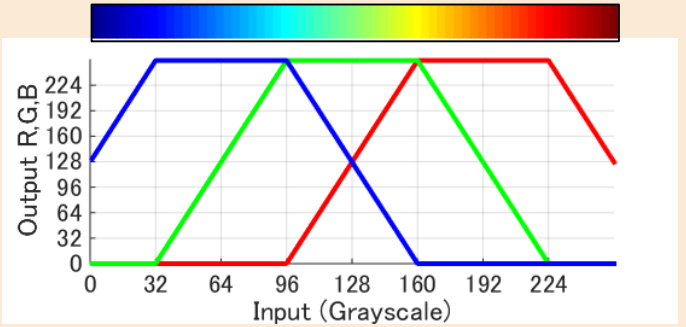


●濃度変換(2)

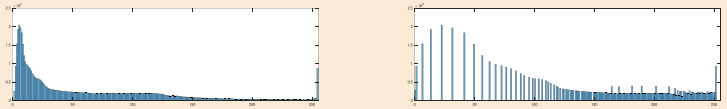


① 疑似カラー

- ➢ グレイスケール値に色(RGB値)を対応付けて表すもの
- ➢ 対応関係を示したもの： カラーマップ

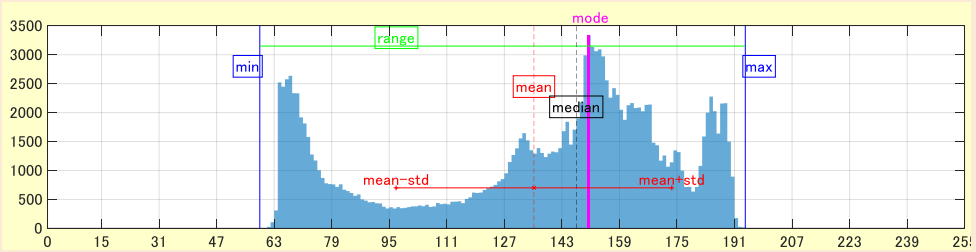


② ヒストグラム平坦化

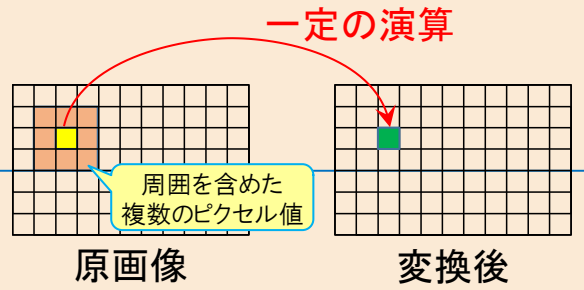


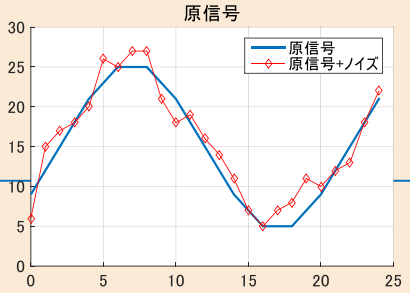
●画像統計量

- ➢最大/最小/最頻
- ➢平均/中央
- ➢範囲/分散/標準偏差



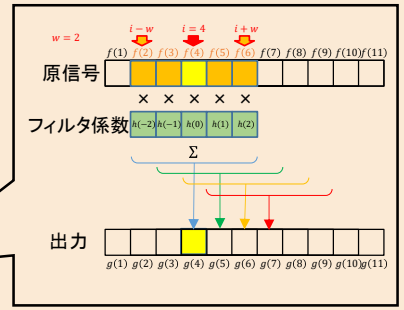
第11回まとめ

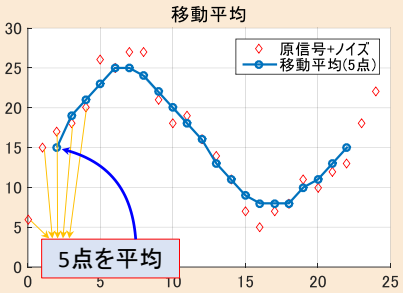




●近傍演算とは？

➢注目ピクセルの近傍(周囲)を含めた
複数のピクセル値を用いて、新たなピクセル値を計算

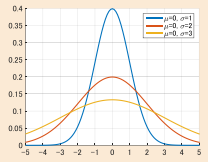




●畳み込み積分

1．1次元信号の畳み込み積分　$g(i) = \sum_{n=-w}^{w} f(i+n)h(n)$

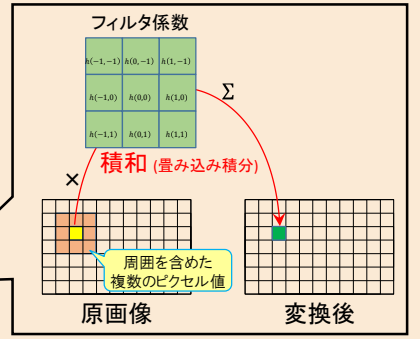
- ➢ 単純移動平均　　$\cdots \quad h(n) = \frac{1}{2w+1}$
- ➢ ガウシアンフィルタ(加重平均の一種)　$\cdots \quad h(n) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{-(n-\mu)^2}{2\sigma^2}}$

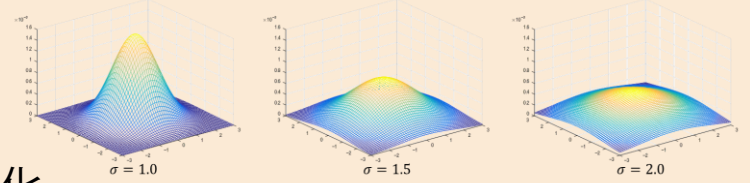


ガウス分布

2．2次元信号(=画像)の畳み込み積分　$g(x,y) = \sum_{n=-w}^{w} \sum_{m=-w}^{w} f(x+m, y+n)h(m,n)$

- ➢ 単純移動平均　　$\cdots \quad h(n) = \frac{1}{(2w+1)^2}$
- ➢ ガウシアンフィルタ(加重平均の一種)　$\cdots \quad h_1(m,n) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{m^2+n^2}{2\sigma^2}\right)}$





2次元正規分布

の二次元正規分布に比例し、フィルタ係数の総和が1.0になるように正規化

(加重)移動平均することで、平滑化された(=高周波成分が低減された)信号になる。
すなわち、一種のLPF(Low Pass Filter: 低域透過フィルタ)として働く。

第12回のまとめ

●輪郭抽出

- 輪郭とは、「ピクセル値が急激に変化しているところ」
- 微分により、輪郭抽出ができる
- 離散信号の微分は、差分を取るだけ → $f'(i) = f(i+1) - f(i)$
- 画像処理としては、近傍演算(畳み込み積分)で実装可能
- 一次微分のPrewittフィルタ/Sobelフィルタ、
  二次微分のLaplacianフィルタなどがよく使われる

●鮮鋭化

- 鮮明(シャープ)な画像とは、
  「輪郭部分でピクセル値が急激に変化している」画像
- 元画像から、二次微分(ラプラシアンフィルタ)した画像を
  減算することで、変化を強調することができる
  ⇒ フィルタ係数同士の演算で得られたフィルタ係数で、
  上記の処理を同時に行うことができる。

$i-1$	i	$i+1$	
0	1	0	$j-1$
1	-4	1	j
0	1	0	$j+1$

4近傍の
Laplacianフィルタ
[※二次微分]

$i-1$	i	$i+1$	
-1	0	1	$j-1$
-1	0	1	j
-1	0	1	$j+1$

横方向に微分
縦方向に平滑化

$i-1$	i	$i+1$	
-1	-1	-1	$j-1$
0	0	0	j
1	1	1	$j+1$

横方向に平滑化
縦方向に微分

Prewittフィルタ
[※一次微分]

$i-1$	i	$i+1$	
-1	0	1	$j-1$
-2	0	2	j
-1	0	1	$j+1$

横方向に微分
縦方向に平滑化

$i-1$	i	$i+1$	
-1	-2	-1	$j-1$
0	0	0	j
1	2	1	$j+1$

横方向に平滑化
縦方向に微分

Sobelフィルタ
[※一次微分]

原画像				ラプラシアンフィルタ				鮮鋭化フィルタ
0	0	0	−	0	1	0	=	0 -1 0
0	1	0		1	-4	1		-1 5 -1
0	0	0		0	1	0		0 -1 0

鮮鋭化フィルタは、フィルタ係数の演算で得ることができる

第13回のまとめ

●畳み込み積分では表現できない近傍演算のひとつとして・・・

「**メディアンフィルタ**」

- 近傍の中央値を、変換後のピクセル値とするもの
- 結果的に、極端に大きな値や、小さな値は無視される

110	120	130	→ 150		
180	190	140			
170	160	150			

10	20	30	→ 150		
220	230	40			
210	200	150			

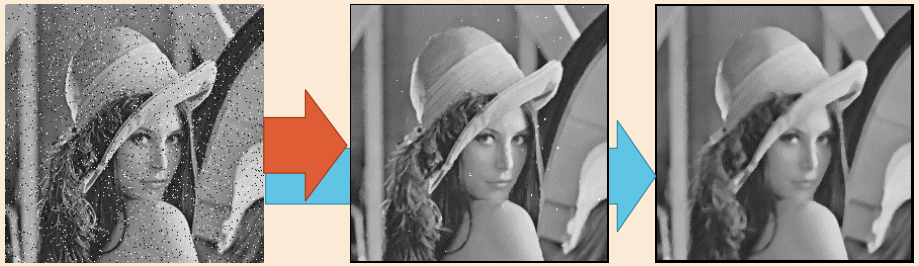
3×3の場合、
ピクセル値の順に並べた際に
5番目に来たピクセルの値にする

0	100	100	→ 150		
200	255	100			
200	200	150			

0	0	0	→ 0		
0	255	0			
0	0	0			

0	0	10	→ 10		
255	255	10			
10	10	10			

- **ごま塩ノイズ**(Salt & Pepper noise / インパルスノイズ(impulse noise)とも)の除去に大きな効果がある。



Median Filter (3x3)　Median Filter (5x5)

【元画像】

Median Filter (3x3)　Median Filter (5x5)

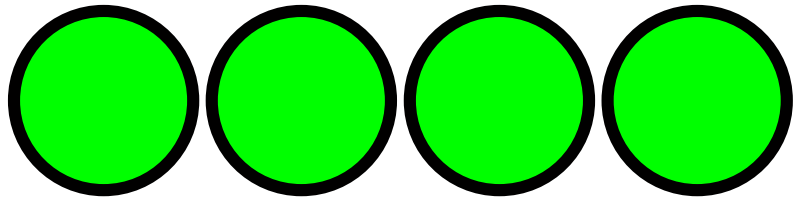
【元画像】

動画像処理

動画とは・・・

●文字通り、「動く画像」が動画

- 静止画の連続を見たとき、ヒトは(実際には動いていない静止画が)
  動画であるように認識する　・・・【**仮現運動**】(かげんうんどう)という
  - パラパラ漫画とか、アニメーションとかが、まさにそれ



- これを利用して、テレビ放送やコンピュータでは、
  通常は　動画を**静止画の連続**　として扱う。

動画に関する用語

- フレーム(frame)　　･･･ 動画を構成する 1コマ（1枚の静止画）のこと
- フレームレート（frame rate)　･･･ 1秒間を何枚のフレームで構成するか
  - fps(frame per second) 単位で表す。
    - 映画　　･･･ 24fps
    - テレビ放送 ･･･ 30fps*　　(*正確には 29.97 fps)
    - アニメーションはほぼ全てが 24fps で制作されている
    - ちなみに、Teams会議の録画は 10fps になっていました。(結構カクカク)
    - 従来は 30fps が標準だったが、最近は 60 fps で撮影/表示できる機材も普及してきている。
      - 例えば、Youtubeは 2014年から 60fps動画 に正式対応
      - 最近はスマホのカメラも、60fps撮影に対応しているものが多い。(※表示も60Hzに対応していないと意味はない…)
      - fpsが高いと、動きがより滑らかに見える(いわゆる、“ヌルヌル動く動画”)
      - 以前は30fpsあれば十分自然に見えるとされていたが、特に大画面では、30fpsと60fpsの違いは分かりやすい

動画像処理

- 動画像は多数の静止画で構成
  = これまでに行ってきたような、静止画に対する画像処理は全て適用可能
- 加えて、複数フレームの情報を用いて、時間軸方向の処理が可能
  - 一般に隣接するフレームは非常によく似た静止画になる
    - その差分を用いたり、対応する点や領域を検出することで、
      カメラの動きの推定や、3次元的な奥行きの推定が出来たりする。
    - 基準とするフレームとの差分情報のみを保持することで、効率の良い動画圧縮が可能となる

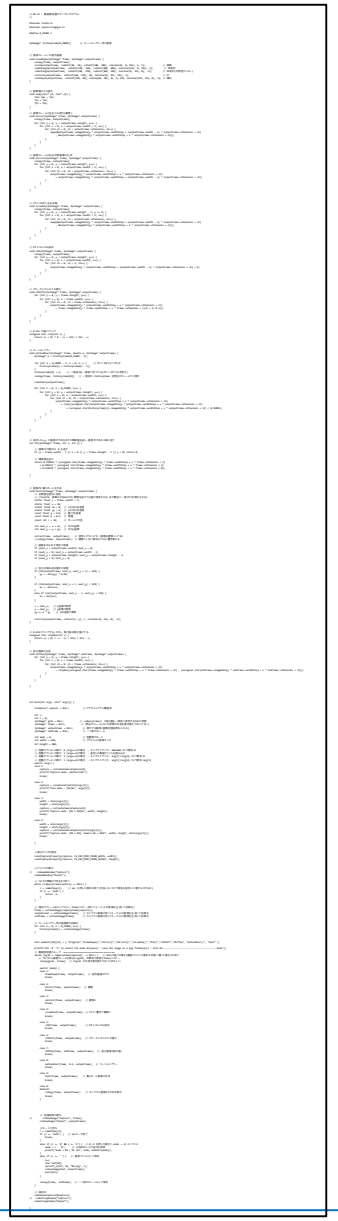
OpenCVでの動画の取り扱い

OpenCVでは、標準でカメラ(webカメラ等)からのリアルタイムでの取り込みと、動画ファイルからの読み込みに対応している。(初期化の部分を変えるだけで、カメラからのリアルタイム処理と動画読み込みの処理を切り替えることも可能)

- カメラの場合は、 cvQueryFrame() 実行時のフレームが取り込まれる。
  - フレームレートは cvQueryFrame() を実行するタイミングに依存する。
  - つまり、cvQueryFrame() を呼ぶ間隔を一定にできれば、固定のフレームレートでの動画像を出力可能だが、画像処理に時間がかかったり、処理の時間が一定ではなかったりするために、一般にはフレームレートは変動する。
  - サンプルのように、とりあえず処理結果を表示するだけであれば、フレームレートの変動はあまり気にしなくて良い。例えば動画から速度を求めたりする場合は、フレーム間の時間を厳密に知る必要がある。
- 動画ファイルの場合は、cvQueryFrame() を実行する毎に、動画ファイルの先頭から1フレームずつ取り込まれる。

あとは、サンプルコードを参考にしてみてください。。。

サンプルコード No.14 (動画像処理)



- 一応完成版です。
  - Webカメラを接続して起動すると、デフォルトで1番目のカメラデバイスから画像が取り込まれます。
    - 起動オプションでカメラの取り込み解像度を指定可能
    - 起動オプションで2番目以降のカメラデバイスを指定可能
  - 生成された exe ファイルに、動画ファイル(mp4等)を Drag&Dropすると、動画の処理になります。

- 0-9 のキーを押すと、処理が切り替わります。
  space キーで、png画像として保存できます。
  esc キーで、終了します。

- 画像処理を行っている各関数は、基本的にこれまでの静止画の処理と同様です。
  - 例えば二値化/グレイスケール化や、近傍処理なども、ほぼそのまま適用できます。
  - 余裕がある人は、適当な処理関数を書き換える等試してみてください。

試験について

主な試験内容

●C言語に関する問題

- ➢基本（多重ループとか、条件式とか）
- ➢関数の書き方、呼び出し方、値渡し/参照渡し
- ➢ポインタや構造体の使い方（書き方）

長々とコードを書かせる問題にはしない(つもり)
基本的に選択問題が多い(はず)（採点も大変になるので…）
※選択問題を多くすると、問題文の文量は多くなるかも。

●画像処理

- ➢ラスタ画像とベクタ画像の違い、用語
- ➢グレイスケール画像と、RGBカラー画像、色について
- ➢グレイスケール化、二値化に関して
- ➢濃度変換 、トーンカーブに関して
- ➢ヒストグラムと各種画像統計量
- ➢近傍演算、畳み込み積分
- ➢授業で扱った各種フィルタについて

例(※これが全てではないし、これをすべて出すわけでもありません)：
- 画像の容量(無圧縮)の計算
- 画像の仕様(bit深度、解像度)が画像処理結果に与える影響
- トーンカーブに対応する処理結果を選択(あるいはその逆)
- トーンカーブ上のどの部分で白飛びや黒つぶれが起きるか
- 画像統計量の計算、ある統計量がどの画像が該当するか
- ヒストグラムに対応する画像を選択(あるいはその逆)
- 近傍演算のフィルタ係数に対応する処理結果の選択
- 各種フィルタに対応する処理結果の選択

例題1

●main()関数内の変数 a と b の和を関数 func() で計算して表示したい。(1)と(2)の部分を埋めなさい。

```
#include <stdio.h>

(1) func(   (2)   ) {
        return x + y;
}

int main(void) {
        int a = 10;
        int b = 20;
        printf("a+b=%d", func(a, b));
        return 0;
}
```

例題2

●正しい出力結果を選べ

①
```
a = 10
*p = 10
```

②
```
a = 20
*p = 10
```

③
```
a = 20
*p = 20
```

④
```
a = 10
*p = 20
```

⑤
```
a = 20
*p = 0
```

⑥
```
a = 20
*p = ???
```
不定

```
#include <stdio.h>

int main(void) {
        int a = 10;
        int *p = NULL;
        p = &a;
        a = 20;

        printf("a = %d¥n", a);
        printf("*p = %d¥n", *p);
        return 0;
}
```

例題3

●右のような構造体を定義した際、
main()関数内で、
data配列の最初の要素の値を表示したいとする。
適切なものを選べ。

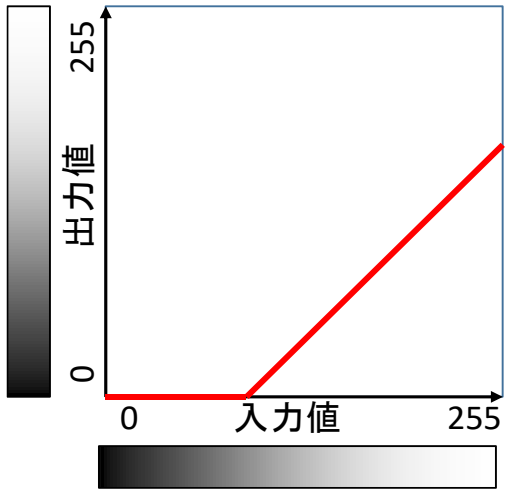
ただし、dataメンバ変数は、init()関数内で、malloc()により適当な長さの配列が確保されているものとする。

① printf("%d", im.data[0]);
② printf("%d", im->data[0]);
③ printf("%d", im[0].data);
④ printf("%d", im[0]->data);
⑤ printf("%d", im[0].data[0]);

```
#include <stdio.h>

typedef struct Image {
        int width;
        int height;
        unsigned char *data;
}

int main(void) {
        Image* im;
        init(im);
        …
        // ここで表示したい。

        return 0;
}
```

例題4

●以下の原画像に、
トーンカーブで示す濃度変換
を行った
結果として
最も適切な
ものを選べ。



①

②

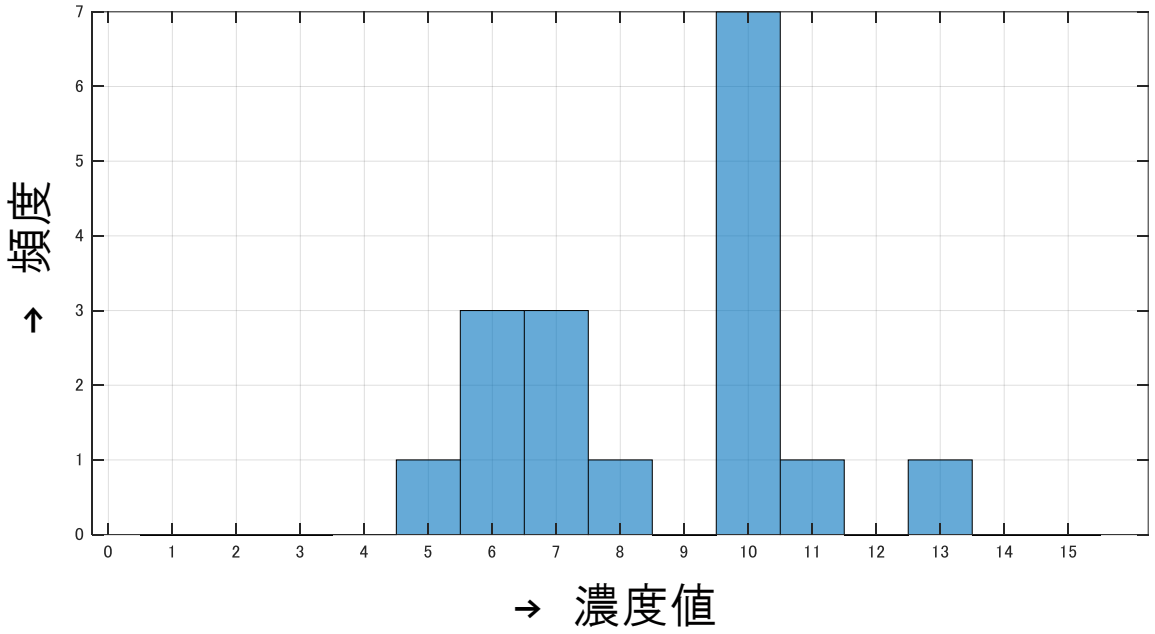
③

④

例題5

●図に示すヒストグラムから、以下の値を答えなさい

① 最小値
② 最大値
③ 最頻値
④ 平均値
⑤ 中央値



例題6

●近傍処理による画像処理結果が
図のようになった。
このとき、使用したフィルタ係数として
最も適切なものを選べ。

画像処理前

画像処理結果

①

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

②

0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04

③

0	0	0
0	1	-1
0	0	0

④

0	1	0
1	-4	1
0	1	0

⑤

1	1	1
1	1	1
1	1	1

⑥

0	0	0
0	1	0
0	0	0

⑦

0	0	0
0	9	0
0	0	0