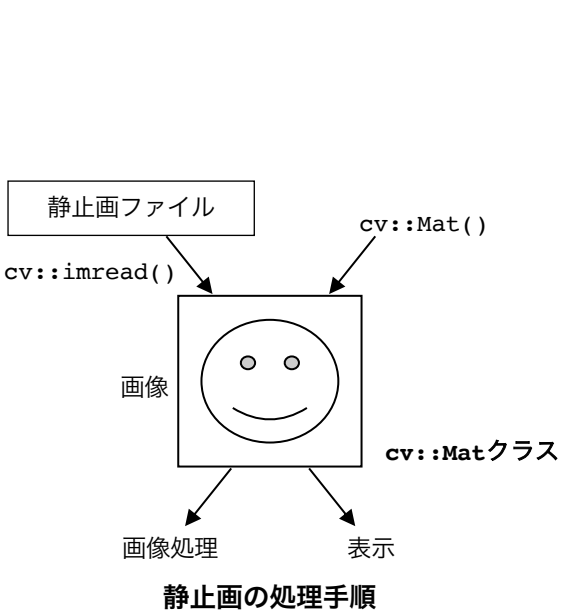
ディジタル映像処理及び演習
第３回

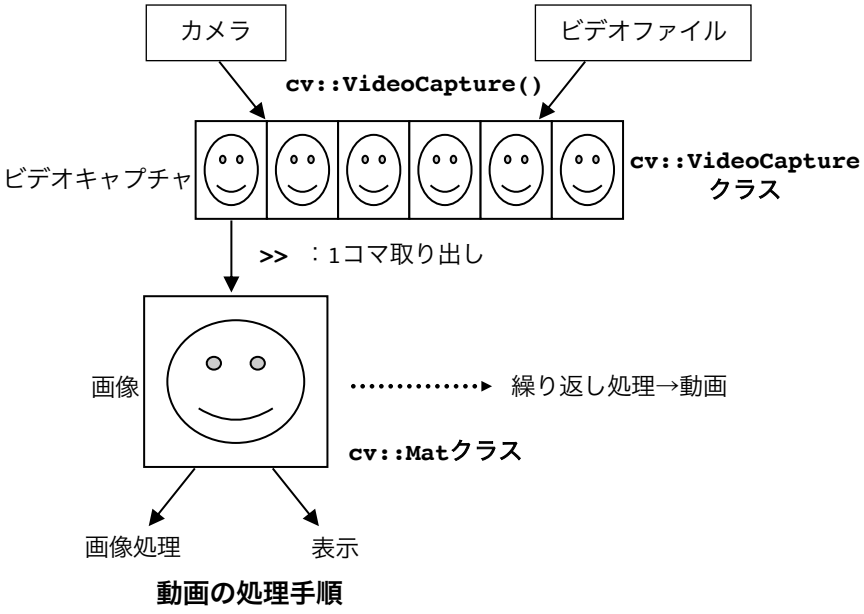
動画像の取り扱いと基本的な処理

○概要

OpenCV では，静止画だけではなく動画も取り扱うことができる．OpenCV で静止画は **cv::Mat クラス**で取り扱って様々な処理や表示を行ってきた．一方，カメラやビデオファイルから得られるビデオキャプチャ（動画）は **cv::VideoCapture クラス**で取り扱う．そして，cv::VideoCapture クラスの動画から 1 フレームずつ静止画として取り出して cv::Mat クラスのインスタンスに格納し，静止画の場合と同様に様々な処理を行う．これを連続的に行うことで動画像の処理や表示を行うことができる．



静止画の処理手順



動画の処理手順

○DIP_03 のソースコード

```
#include <iostream>  //入出力関連ヘッダ
#include <opencv2/opencv.hpp>  //OpenCV 関連ヘッダ

int main (int argc, const char* argv[]) {

    //①カメラの初期化
    cv::VideoCapture capture(0);  //カメラ 0 番をオープン
    //カメラがオープンできたかどうかをチェック
    if (capture.isOpened()==0) {
        printf("Camera not found\n");
        return -1;
    }

    //②画像格納用インスタンス準備
    int width = 640, height = 360;  //処理画像サイズ
    cv::Mat captureImage;  //キャプチャ用
    cv::Mat frameImage = cv::Mat(cv::Size(width, height), CV_8UC3);  //処理用
    cv::Mat grayImage;  //処理用

    //③ウィンドウの生成と移動
    cv::namedWindow("Frame");
    cv::moveWindow("Frame", 0,0);
    cv::namedWindow("Result");
    cv::moveWindow("Result", 0,height);

    //④カメラから 1 フレーム読み込んで captureImage に格納
    capture >> captureImage;

    //⑤captureImage を frameImage に合わせてサイズ変換して格納
    cv::resize(captureImage, frameImage, frameImage.size());

    //⑥画像処理
    cv::cvtColor(frameImage, grayImage, cv::COLOR_BGR2GRAY);
```

```
    //⑦ウィンドウへの画像の表示
    cv::imshow("Frame", frameImage);
    cv::imshow("Result", grayImage);

    //⑧キー入力待ち
    cv::waitKey(0);

    return 0;
}
```

○ソースコードおよび使用した関数の説明

手順①では，カメラの初期化を行っている．まず，cv::VideoCaptureクラスのインスタンス("capture")を生成することでカメラをオープンしている．

cv::VideoCapture クラス
カメラやビデオファイルからキャプチャするクラス．主なメンバは以下の通り．

```
VideoCapture(int device);                  //カメラをオープンするコンストラクタ
VideoCapture(const string& filename);      //ビデオファイルをオープンするコンストラクタ
bool isOpened();
```

カメラ/ビデオがオープンできたかどうか調べるメソッド．

```
bool set(int propid, double value);
```

ビデオキャプチャの設定を行う関数．引数 propidは設定対象で，cv::CAP_PROP_FRAME_WIDTH（画像取得の幅），cv::CAP_PROP_FRAME_HEIGHT（画像取得の高さ）など．引数 valueは設定値．

サンプルプログラムではインデックス 0 のカメラ（通常は PC 内蔵のカメラ）をオープンを試みている．そして，カメラがオープンできたかどうかを isOpened()メソッドで調べている．オープンに失敗した場合にはメッセージを出して終了する．

動画像はカメラだけでなく，ほぼ同様の手法でビデオファイルからも取得できる．例えば，ビデオファイル"doga.mov"からcv::VideoCaptureクラスのインスタンス("capture2")を取得するには，以下のように記述すればよい．

```
cv::VideoCapture capture2("doga.mov");
```

手順②では，画像格納用の cv::Mat クラスのインスタンスを生成している．captureImageは動画から 1 フレームをキャプチャして一時的に格納するための画像である．frameImageはキャプチャ画像を処理用にサイズ変更して格納するための画像である．そのため，画像サイズやチャンネル数などを事前に決定している．grayImageは処理用画像をさらにグレースケール変換して格納するための画像である．

手順③では，画像表示用の 2 つのウィンドウを生成している．

手順④では，カメラから 1 フレームを読み込む．具体的には，**入力演算子 >>** を用いて，ビデオキャプチャ capture0から 1 フレーム読み込んで，画像 captureImageに格納している． captureImageは BGR 形式の 3 チャンネルカラー画像となる．

手順⑤では，画像処理の負荷を調整するために画像サイズの変更を行っている．具体的には，cv:resize()関数を用いて，captureImageをサイズの小さい frameImageに変換して格納している．

cv::resize(cv::Mat src, cv::Mat dst, cv::Size size);
画像サイズを変更する関数．引数 srcは入力画像．引数 dstは出力画像で，srcと同じデプスやチャンネル数である必要がある．引数 sizeは出力画像のサイズ．

手順⑥では，cv:cvtColor()関数を用いて，BGR カラー画像である frameImageをグレースケール画像に変換して grayImageに格納している．

手順⑦では，2 つのウィンドウにそれぞれ画像を表示している．

手順⑧では，キー入力の待機を待機して，終了処理を行っている．

○動画像の処理と表示

先のプログラムでは，起動直後にカメラのビデオキャプチャから 1 フレームだけを取り込んで，サイズを変換したりグレースケール画像に変換したりして，ウィンドウに表示していた．そのためウィンドウに表示されるのは静止画であった．

動画像を処理して表示するには，「カメラのビデオキャプチャから画像を取り込み→画像処理→ウィンドウに表示→少し待機」という一連の手順を繰り返し行う必要がある．つまり，無限ループ（while(1)）に入って，

・手順④：「ビデオキャプチャから 1 フレームを取り込んで，captureImageに格納」

・手順⑤⑥：「captureImageをサイズ変換して frameImageに格納してから，グレースケール変換して resultImageに格納」

・手順⑦：「2 つのウィンドウにそれぞれの画像を表示」

・手順⑧：「ビデオキャプチャのフレームレートに応じて少し待機」

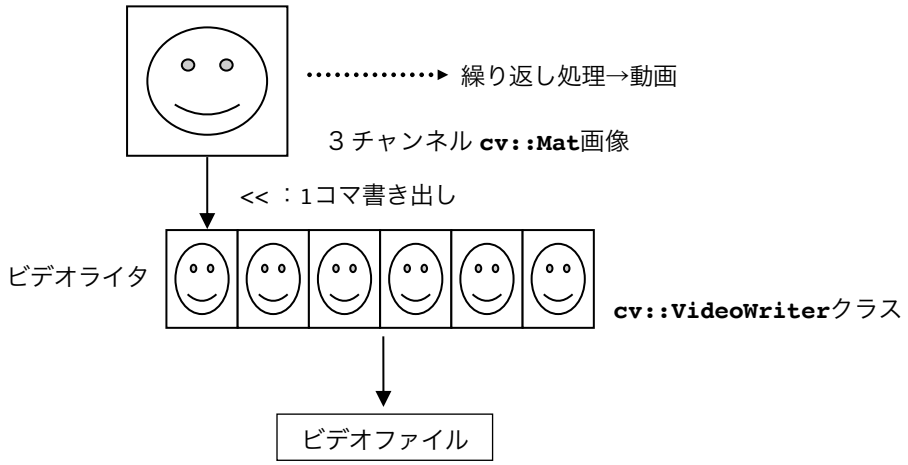
を繰り返し行えばよい．なお，手順④ではフレームの取り込みに失敗したとき，手順⑧では特定のキーが入力されたとき，動画像処理無限ループから脱出するようにしている．

以下にプログラムの修正例を示す．

```
#include <iostream>  //入出力関連ヘッダ
#include <opencv2/opencv.hpp>  //OpenCV 関連ヘッダ

int main (int argc, const char* argv[]) {

    //①カメラの初期化
    cv::VideoCapture capture(0);  //カメラ 0 番をオープン
    //カメラがオープンできたかどうかをチェック
    if (capture.isOpened()==0) {
        printf("Camera not found\n");
        return -1;
    }

    //②画像格納用インスタンス準備
    int width = 640, height = 360;  //処理画像サイズ
    cv::Mat captureImage;  //キャプチャ用
    cv::Mat frameImage = cv::Mat(cv::Size(width, height), CV_8UC3);  //処理用
    cv::Mat grayImage;  //処理用

    //③ウィンドウの生成と移動
    cv::namedWindow("Frame");
    cv::moveWindow("Frame", 0,0);
    cv::namedWindow("Result");
    cv::moveWindow("Result", 0,height);

    //動画像処理無限ループ
    while (1) {
        //④カメラから 1 フレーム読み込んで captureImage に格納
        capture >> captureImage;
        if (captureImage.data==0) break;  //フレーム読み込みに失敗した場合には無限ループから脱出

        //⑤captureImage を frameImage に合わせてサイズ変換して格納
        cv::resize(captureImage, frameImage, frameImage.size());

        //⑥画像処理
        cv::cvtColor(frameImage, grayImage, cv::COLOR_BGR2GRAY);

        //⑦ウィンドウへの画像の表示
        cv::imshow("Frame", frameImage);
        cv::imshow("Result", grayImage);

        //⑧'キー入力待ち
        char key = cv::waitKey(20);
        if (key=='q') break;
    }

    return 0;
}
```

◯動画像の保存

OpenCV では動画像をビデオファイルとして保存することができる．ビデオファイルを作成するには，**cv::VideoWriter クラス**のインスタンスのビデオライタを用いる．そして**出力演算子 <<** を用いてビデオライタの各フレームに **cv::Mat**画像を出力していけばよい．このとき，出力する画像は基本的には **3 チャンネル**である必要がある．



ビデオライタは以下の書式で生成する．

VideoWriter::VideoWriter(const string& filename, int fourcc, double fps, Size frameSize);
ビデオライタのコンストラクタ関数．引数 filenameは動画ファイル名，引数 fourccは動画コーデックで，例えば cv::VideoWriter::fourcc('P','I','M','1')，cv::VideoWriter::fourcc('M','P','4','V')と記述すればそれぞれ，MPEG1(.mpg)，MP4(.mp4)となる．引数 frameSizeはフレームサイズ．

```
---------------------------------------------------------------------------------------------------
#include <iostream>  //入出力関連ヘッダ
#include <opencv2/opencv.hpp>  //OpenCV 関連ヘッダ

int main (int argc, const char* argv[]) {

    //①カメラの初期化
    cv::VideoCapture capture("dance1.mov");  //カメラ 0 番をオープン
    //カメラがオープンできたかどうかをチェック
    if (capture.isOpened()==0) {
        printf("Camera not found\n");
        return -1;
    }

    //②画像格納用インスタンス準備
    int width = 640, height = 360;  //処理画像サイズ
    cv::Mat captureImage;  //キャプチャ用
    cv::Mat frameImage = cv::Mat(cv::Size(width, height), CV_8UC3);  //処理用
    cv::Mat grayImage;  //処理用
    cv::Mat recImage = cv::Mat(cv::Size(width, height), CV_8UC3);  //ファイル出力用

    //③ウィンドウの生成と移動
    cv::namedWindow("Frame");
    cv::moveWindow("Frame", 0,0);
    cv::namedWindow("Result");
    cv::moveWindow("Result", 0,height);

    //(X)ビデオライタ生成
    cv::VideoWriter rec("rec.mpg", cv::VideoWriter::fourcc('P','I','M','1'), 30, recImage.size());

    //動画像処理無限ループ
    while (1) {
        //④カメラから 1 フレーム読み込んで captureImage に格納
        capture >> captureImage;
        if (captureImage.data==0) break;  //フレーム読み込みに失敗した場合には無限ループから脱出

        //⑤captureImage を frameImage に合わせてサイズ変換して格納
        cv::resize(captureImage, frameImage, frameImage.size());
```

```
        //⑥画像処理
        cv::cvtColor(frameImage, grayImage, cv::COLOR_BGR2GRAY);

        //⑦ウィンドウへの画像の表示
        cv::imshow("Frame", frameImage);
        cv::imshow("Result", grayImage);

        //(Y)動画ファイル書き出し
        cv::cvtColor(grayImage, recImage, cv::COLOR_GRAY2BGR);  //動画用 3 チャンネル画像生成
        rec << recImage;  //ビデオライタに画像出力

        //⑧'キー入力待ち
        char key = cv::waitKey(30);
        if (key=='q') break;
    }

    return 0;
}
```

演習課題

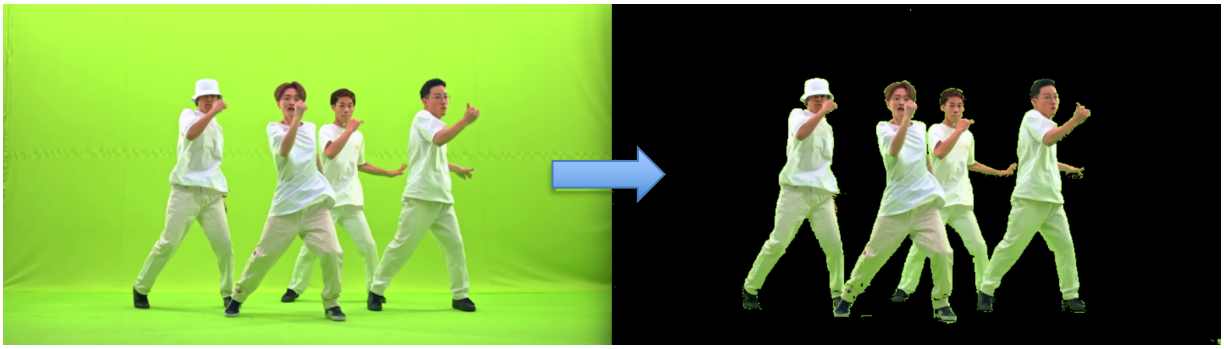
1．Web カメラで読み込んだ自分の顔の動画を 2 値化変換して，ビデオファイルとして書き出しなさい．ビデオファイルの時間は 10 秒程度としなさい．再生できることを確認したビデオファイルを提出しなさい．

<ヒント>
動画像処理無限ループ内において，ビデオの各フレームのグレースケール画像 `grayImage`を BGR 画像に変換する前に 2 値化すればよい．2 値化は第 1 回演習で使用した `cv::threshold()`関数を用いる．しきい値を適切に設定して，顔がある程度認識できるようにすること．
なお，見かけ上はグレースケール画像であっても，ビデオ書き出し用画像は 3 チャンネルカラー画像にすること．

2．Moodle に用意したビデオファイル"dance.mov"について，ダンサーだけをできるだけ残して，背景部分を削除して黒くしたビデオを生成して，ビデオファイルとして書き出しなさい．再生できることを確認したビデオファイルを提出しなさい．
なおビデオファイルの読み込みは，" dance.mov"ファイルをプロジェクトフォルダに置いてから，手順①を以下のように修正すればできる．

```
cv::VideoCapture capture(0);
        ↓
cv::VideoCapture capture("dance.mov");
```

<ヒント>
第 2 回の課題 2 の動画応用．`cv::cvtColor()`関数で `frameImage`を一旦 HSV 画像に変換してから，1 画素ごとに H 成分(色相)，S 成分(彩度)，V 成分(明度)を調べて，背景でない部分に該当する値を持つ画素を抽出していく．
なお，HSV 画像で処理した場合には，正しい色のビデオの生成とファイルの書き出しのため，<<演算子でフレーム画像を書き込む際に `cv::cvtColor()`関数で HSV 画像から BGR 画像に戻す必要がある．



3．課題 2 で黒くした背景にビデオファイル"landscape.mov"の動画を埋め込みなさい．再生できることを確認したビデオファイルを提出しなさい．

<ヒント>
ビデオファイル"landscape.mov"を読み込むために `capture2`を用意するなどして，2 つのビデオを同時に読み込む処理を行う．