OpenCV による画像処理

目次

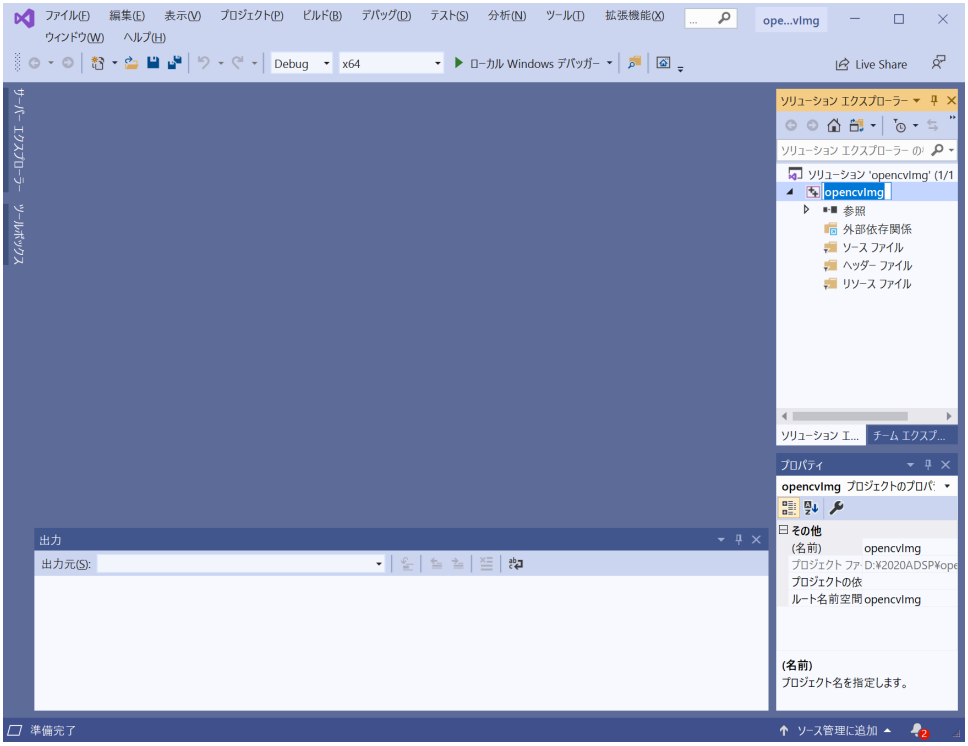
１．簡単な画像入力と表示 2
２．画像に文字描画と画像の出力 6
３．移動平均フィルタ処理 8
　課題：メディアンフィルタ 9
４．顔抽出 10
　課題：顔検出 11
参考文献 12
付録 1：VisualStudio2019 における OpenCV 使用環境の設定 13
付録 2：OpenCV の顔認識分類器 18

１．簡単な画像入力と表示

(1) プロジェクト作成

Visual Studio 2019 を起動し、「新しいプロジェクトの作成」より「空のプロジェクト」を作成

場所：以前作った「ADSP」に、プロジェクト名「opencvImg」



(2) opencv 環境設定

① メニューの「表示」の「プロパティ　ページ」をクリックし、「opencvImg　プロパティ　ページ」を開く

もしくは、メニューの「プロジェクト」の「プロパティ」をクリックし、「opencvImg　プロパティ　ページ」を開く

② 「構成」を"全ての構成"に、「プラットフォーム」を"x64"に設定

③ 構成のプロパティより、下記のように設定：

(a) VC++　ディレクトリ　→　インクルードディレクトリ　に下記を追加

C:¥opencv¥build¥include

（インクルードディレクトリの右側にある「V」をクリックし、＜編集＞を選び、上記を追加）

(b) VC++　ディレクトリ　→　ライブラリディレクトリ　に下記を追加

C:¥opencv¥build¥x64¥vc15¥lib

(c) リンカー　→　入力　→　追加の依存ファイル　に下記を追加

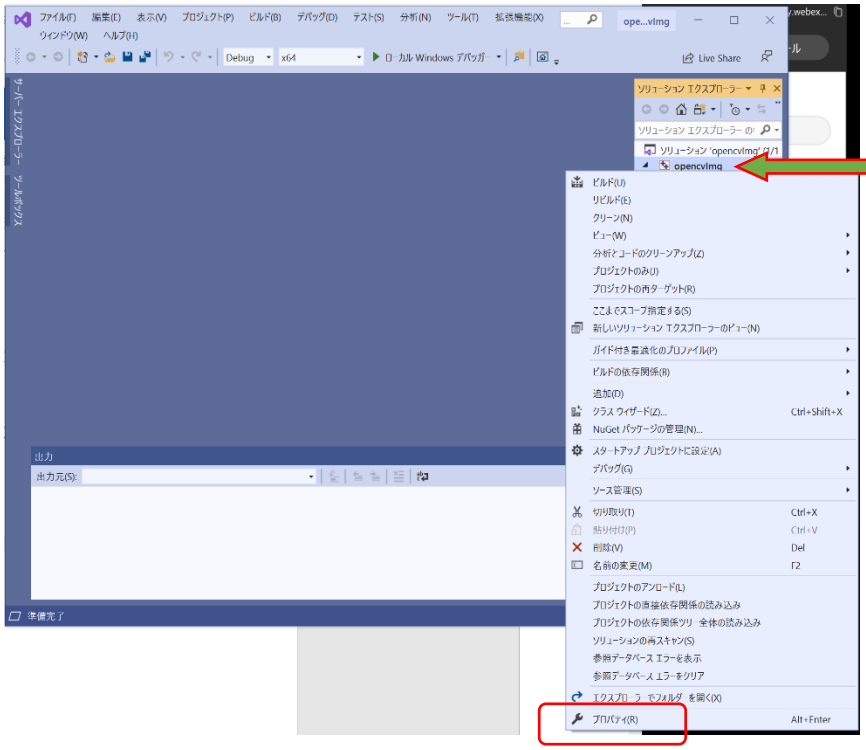
opencv_world460d.lib

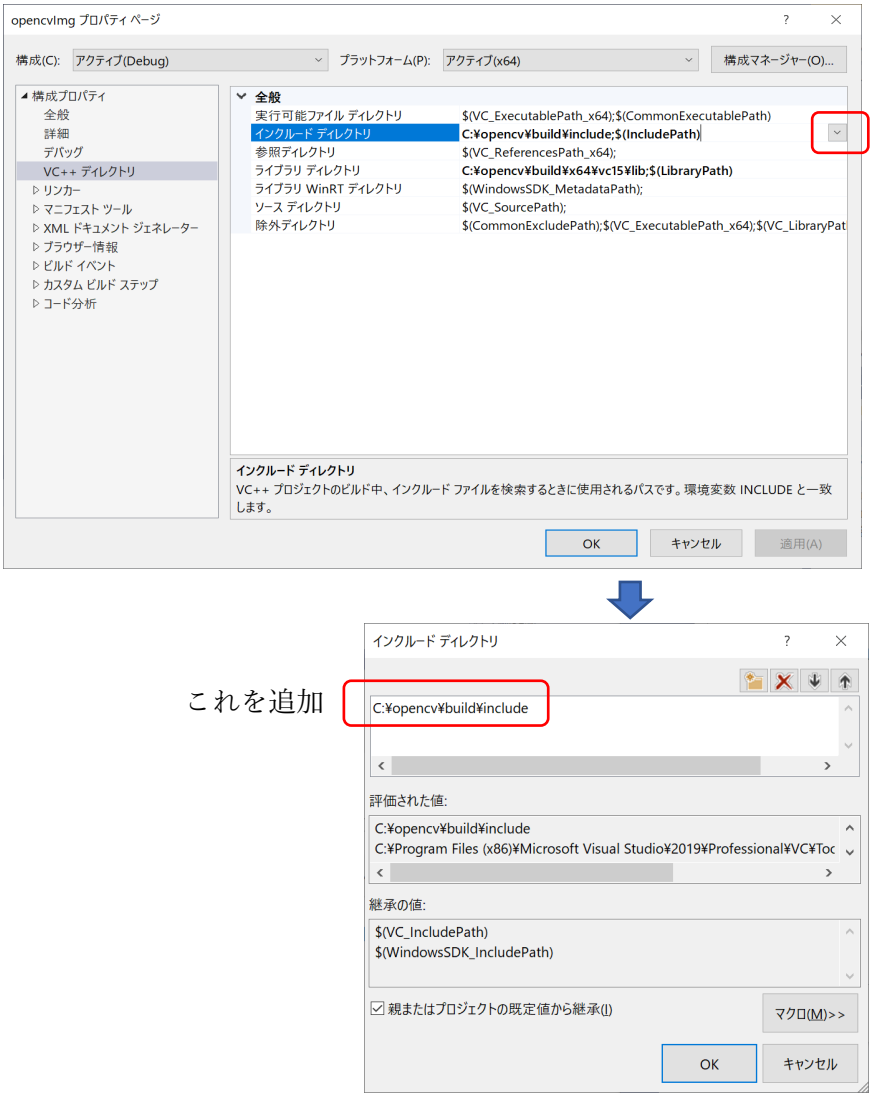
（今回は Debug の例、Release の場合には opencv_world460.lib を追加）

（バージョンにより、460 を 440、411 などに変更する）

(d) デバッグ　＞　環境　に下記を追加

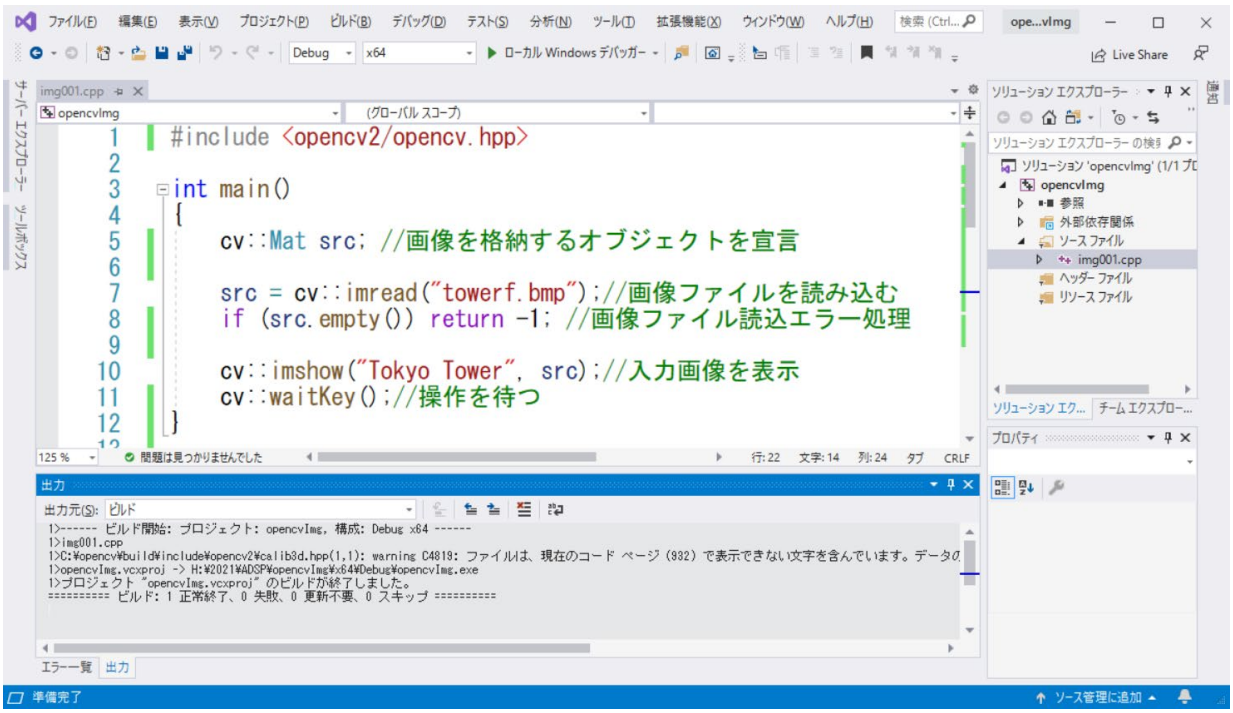
PATH=C:¥opencv¥build¥x64¥vc15¥bin


ここを右クリック
opencvImg
ビルド(U)
リビルド(E)
クリーン(N)
プロジェクトのみ(J)
プロジェクトの再ターゲット(R)
新しいソリューション エクスプローラーのビュー(N)
ビルドの依存関係(B)
追加(D)
クラス ウィザード(Z)... Ctrl+Shift+X
NuGet パッケージの管理(N)...
デバッグ(G)
ソース管理(S)
切り取り(T) Ctrl+X
削除(V) Del
名前の変更(M) F2
プロジェクトのアンロード(L)
ソリューションの再スキャン(S)
プロパティ(R) Alt+Enter
出力
出力元(S):
準備完了

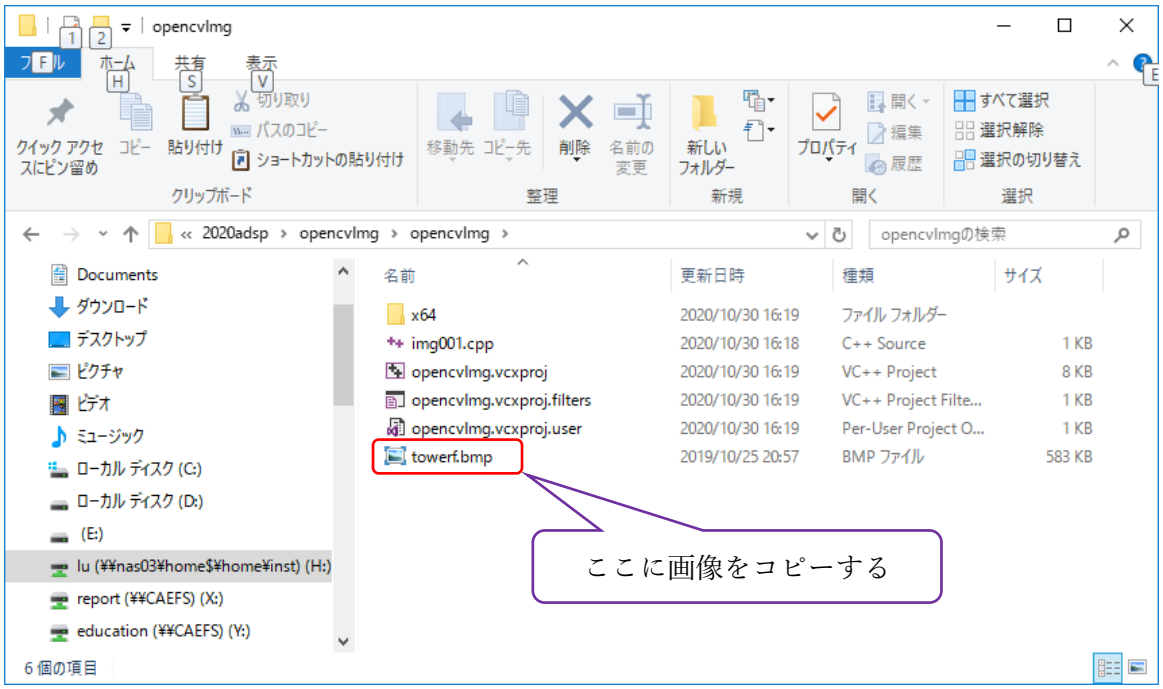


(3) ソースファイルを追加

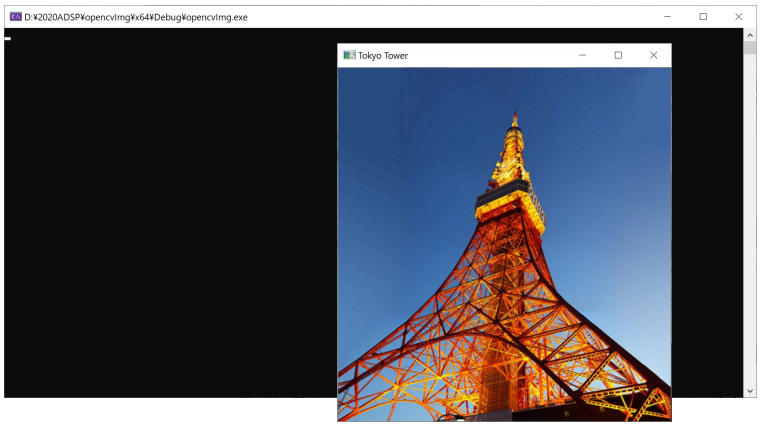
ソースファイルに CPP ソースファイル「img001.cpp」を新規追加



(4) 処理用画像「towerf.bmp」をプロジェクタのフォルダにコピー



(5) ビルドして実行（デバッグなしで開始（H））して、下記のように画像が表示される。
任意のキーを押したら画像の表示が終了。



これで、OpenCV による簡単な画像入出力が成功した！

ソースコード説明：

cv::Mat　　画像用オブジェクト

cv::imread(　)　　画像ファイルを読み込む

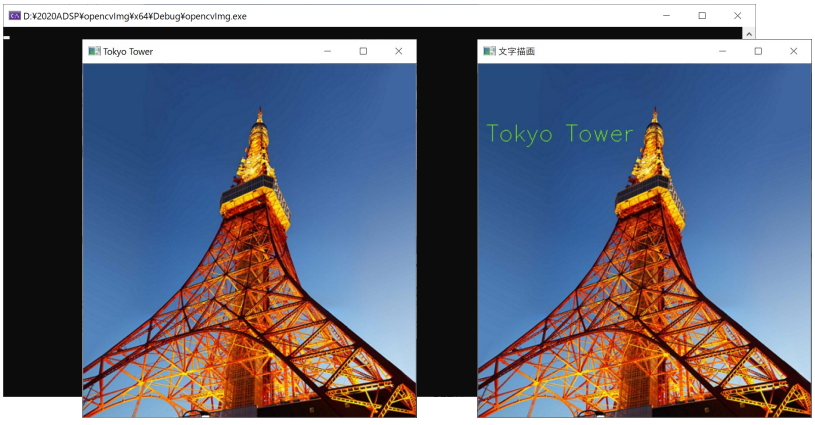
cv::imshow(　)　　モニターに画像表示

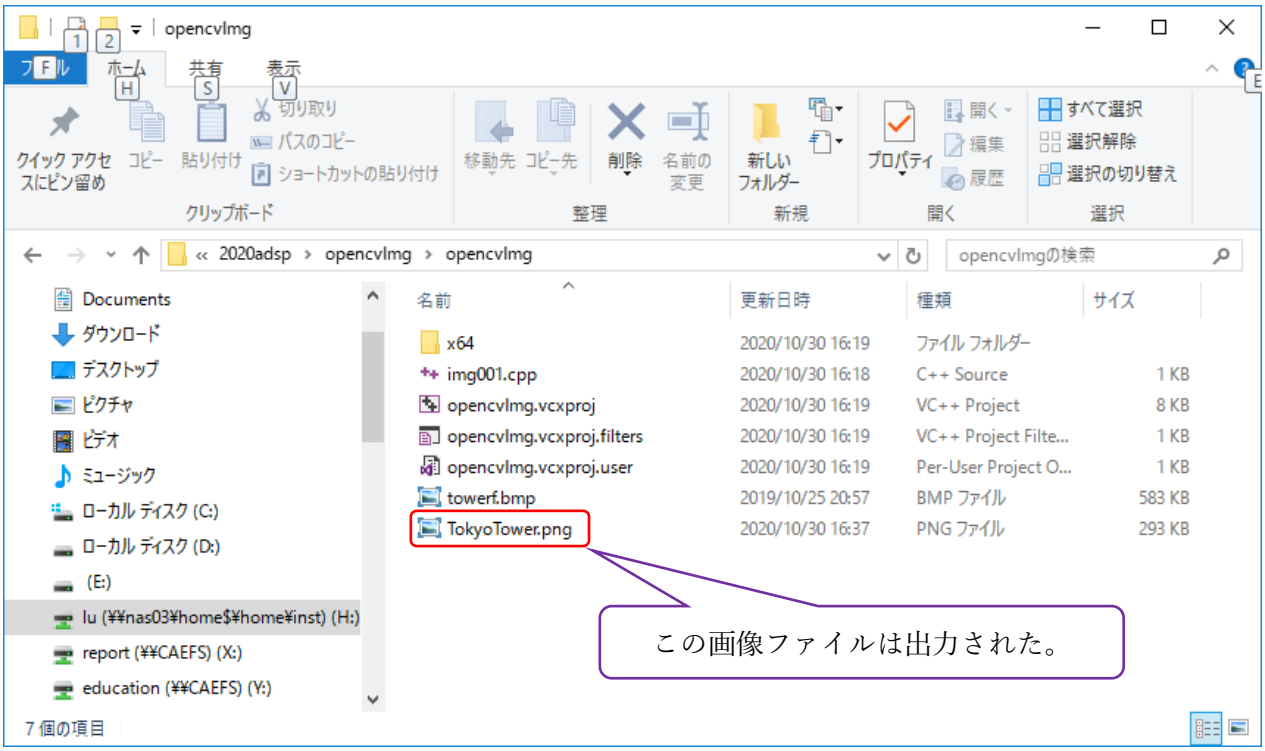
詳細は OpenCV の公式 HP を参照ください。

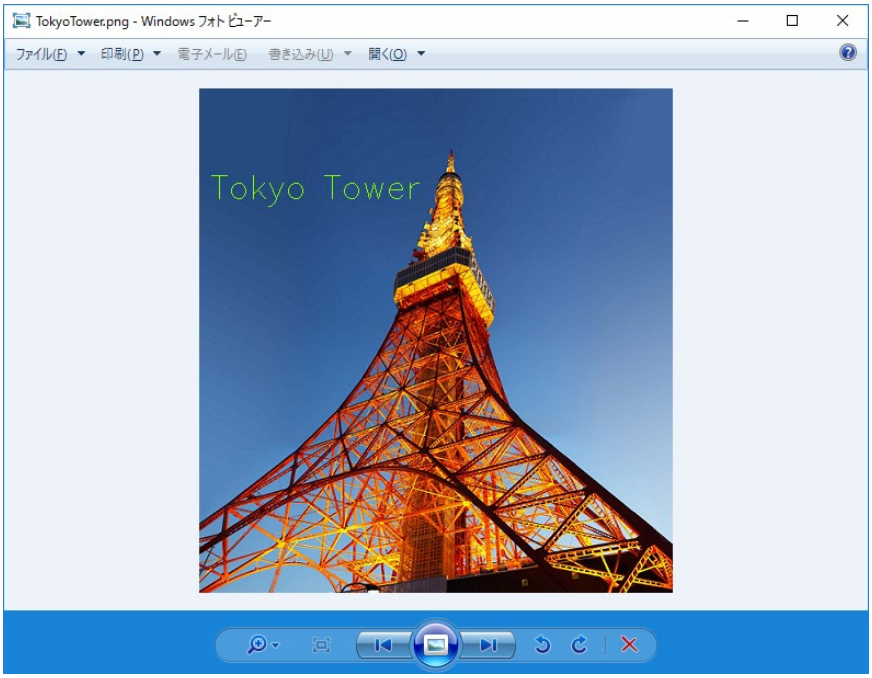
２．画像に文字描画と画像の出力

ソースファイル「img001.cpp」に下記のようにコードを追加し、実行せよ。下記の画像が表示され、画像ファイル「TokyoTower.png」が出力される。

```cpp
#include <opencv2/opencv.hpp>

int main()
{
    cv::Mat src; //画像を格納するオブジェクトを宣言

    src = cv::imread("towerf.bmp");//画像ファイルを読み込む
    if (src.empty()) return -1; //画像ファイル読込エラー処理

    cv::imshow("Tokyo Tower", src);//入力画像を表示

    //テキスト描画
    cv::String text = "Tokyo Tower";//描画用文字列
    cv::Point org(10, 100);//描画の始点座標
    int fontFace = cv::FONT_HERSHEY_SIMPLEX;//描画フォント指定
    double fontScale = 1.0;//描画文字サイズ指定
    cv::Scalar color(0, 255, 127);//描画色指定、B,G,Rの色強度
    cv::putText(src, text, org, fontFace, fontScale, color);//描画

    // 画像を出力する
    cv::String path = "TokyoTower.png";//出力ファイル名
    cv::imwrite(path, src);

    cv::imshow("文字描画", src);//入力画像に文字列を表示

    cv::waitKey();//操作を待つ
}
```



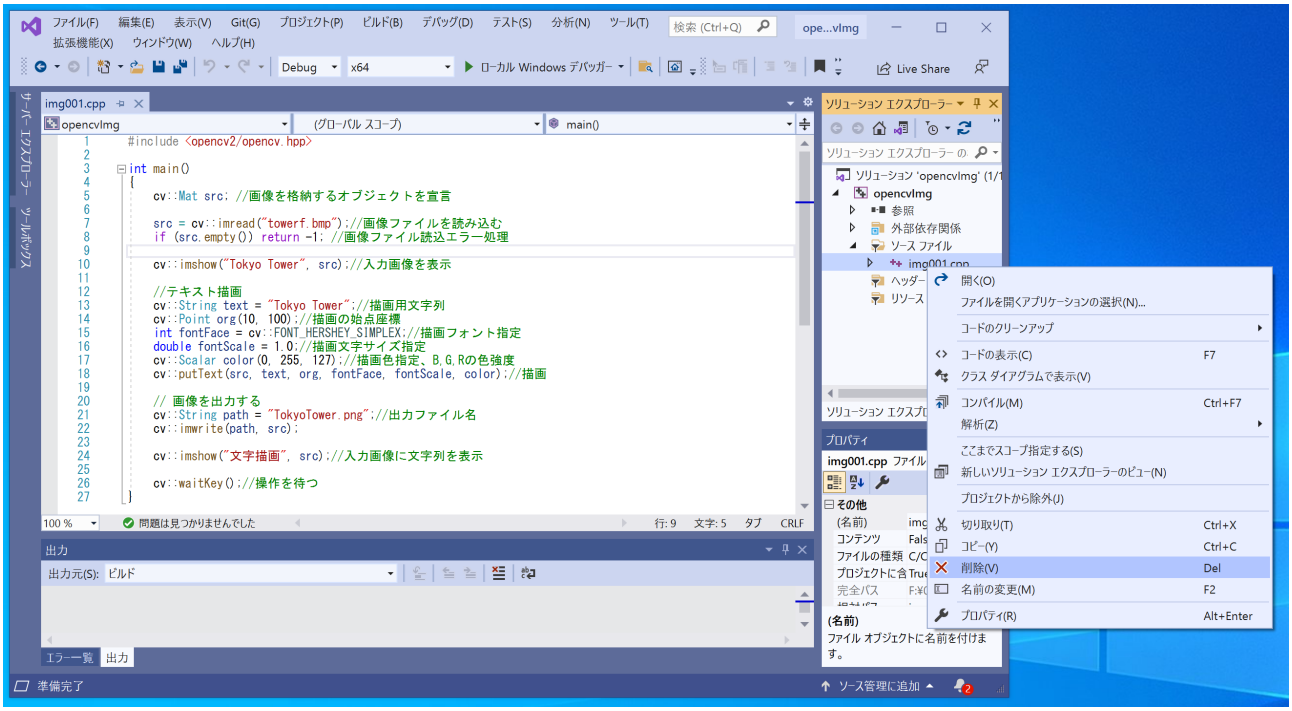

opencvImg
ファイル
ホーム
共有
表示
クイック アクセスにピン留め
コピー
貼り付け
切り取り
パスのコピー
ショートカットの貼り付け
クリップボード
移動先
コピー先
削除
名前の変更
整理
新しいフォルダー
新規
プロパティ
開く
編集
履歴
開く
すべて選択
選択解除
選択の切り替え
選択
« 2020adsp › opencvImg › opencvImg
opencvImgの検索
Documents
ダウンロード
デスクトップ
ピクチャ
ビデオ
ミュージック
ローカル ディスク (C:)
ローカル ディスク (D:)
(E:)
lu (¥¥nas03¥home$¥home¥inst) (H:)
report (¥¥CAEFS) (X:)
education (¥¥CAEFS) (Y:)
名前
更新日時
種類
サイズ
x64 2020/10/30 16:19 ファイル フォルダー
img001.cpp 2020/10/30 16:18 C++ Source 1 KB
opencvImg.vcxproj 2020/10/30 16:19 VC++ Project 8 KB
opencvImg.vcxproj.filters 2020/10/30 16:19 VC++ Project Filte... 1 KB
opencvImg.vcxproj.user 2020/10/30 16:19 Per-User Project O... 1 KB
towerf.bmp 2019/10/25 20:57 BMP ファイル 583 KB
TokyoTower.png 2020/10/30 16:37 PNG ファイル 293 KB
この画像ファイルは出力された。
7 個の項目

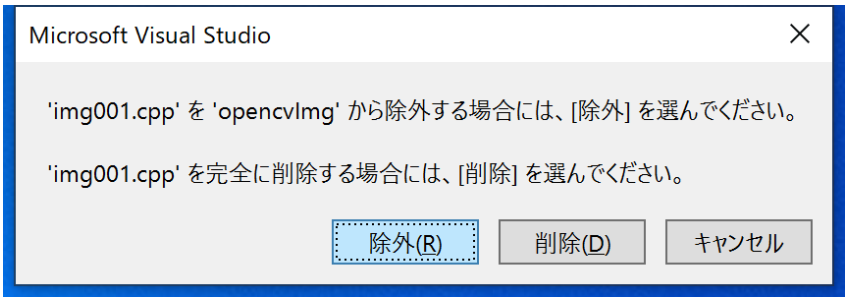

TokyoTower.png - Windows フォト ビューアー
ファイル(F)
印刷(P)
電子メール(E)
書き込み(U)
開く(O)
Tokyo Tower

3．移動平均フィルタ処理

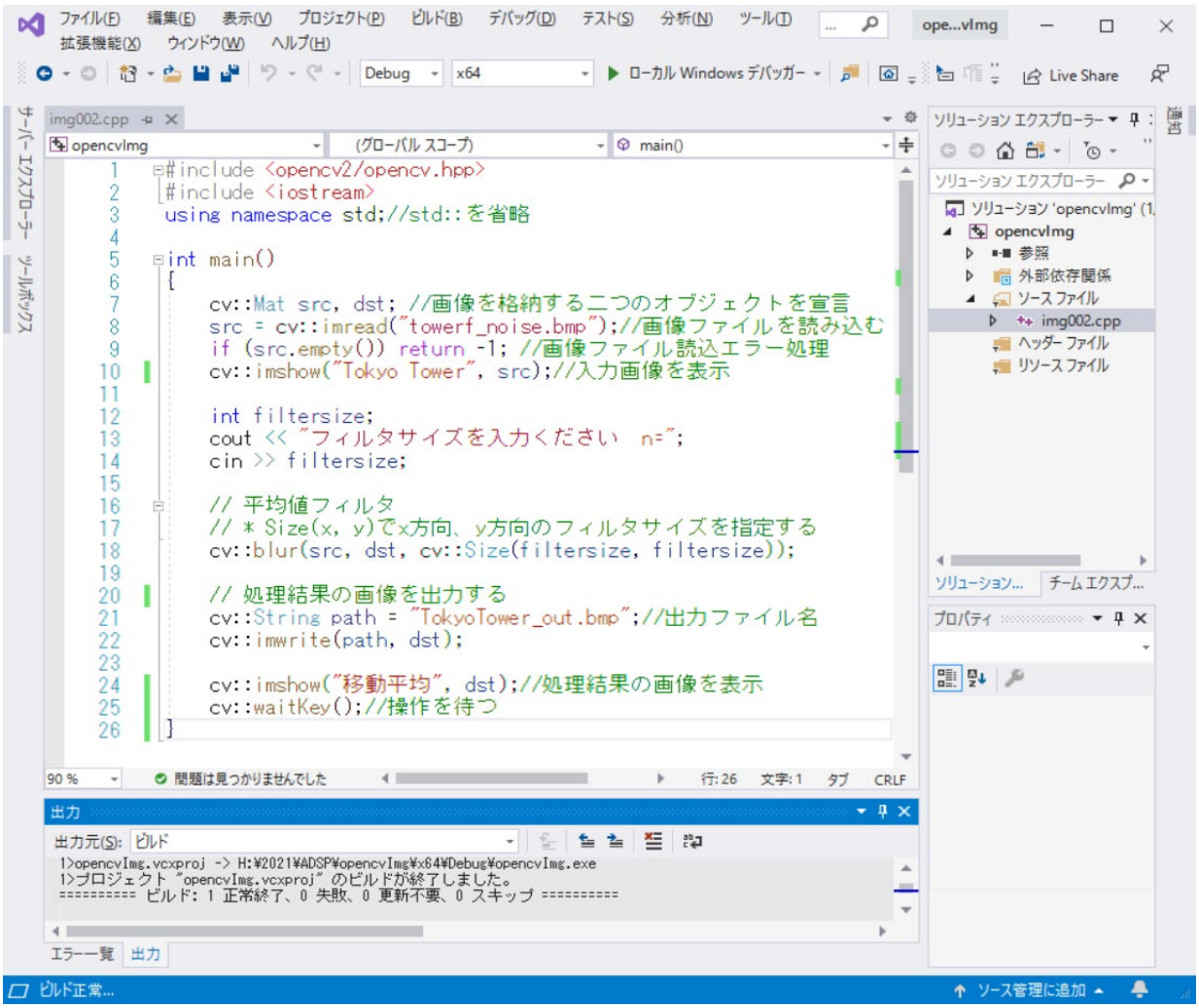
ノイズのある画像ファイル「towerf_noise.bmp」をプロジェクトフォルダに入れる。

ソースファイルの「img001.cpp」を右クリックし、「削除」→「除外」する。

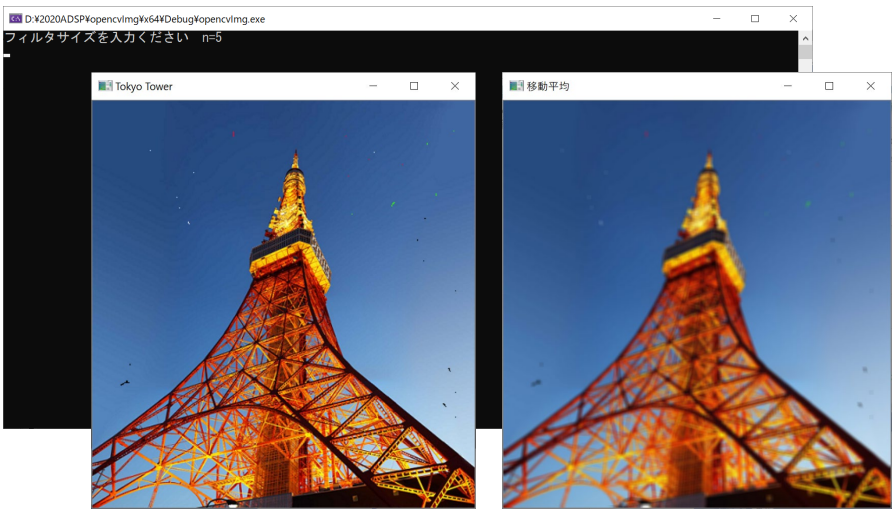




下記のように、ソースファイルに「img002.cpp」を新規追加し、実行する。(注意:include の追加が必要)



いろいろなフィルタサイズを入力し、出力を比較せよ。



課題：メディアンフィルタ

メディアンフィルタ処理を行い、フィルタサイズをそれぞれ1、3、5、7、9にして、結果を考察せよ。

メディアンフィルタ処理には、下記の関数を使うことが可能である。

cv::medianBlur(src_img, dst_img, size)

４．顔抽出

プロジェクトの C ソースを「imag001.cpp」や「imag002.cpp」より「face001.cpp」に差し替え（下記を参照）、実行結果を考察せよ。

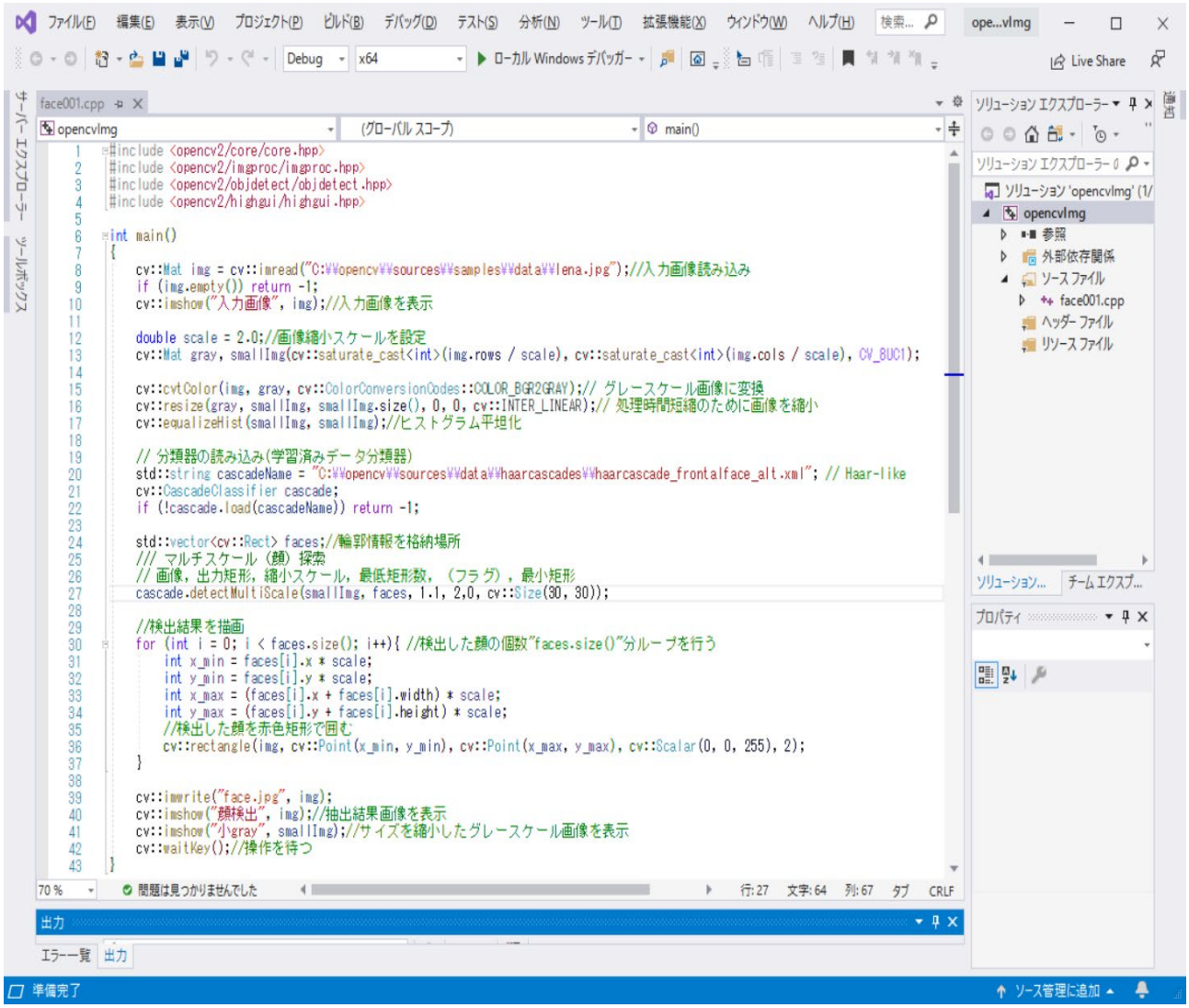
(1)　画像読込

(2)　画像縮小（処理時間短縮のための操作。この部分を省略しても構わない）

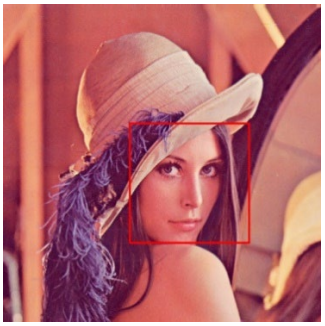
(3)　分類器読込

(4)　顔検出

(5)　結果画像出力、描画



元画像



顔抽出結果

課題：顔検出

(1) 入力画像を縮小せずそのまま処理するプログラムを作れ。
　　注:サンプルの scale=1 のように、スケール値の設定変更ではなく、scale 関連の操作を全部なくすこと。

(2) 自分の顔写真など他の画像を用い、処理せよ。

(3) 複数の顔がある画像を用い、処理せよ。

(4) 異なる分類器を用い、顔や目などの抽出処理をせよ。

参考文献

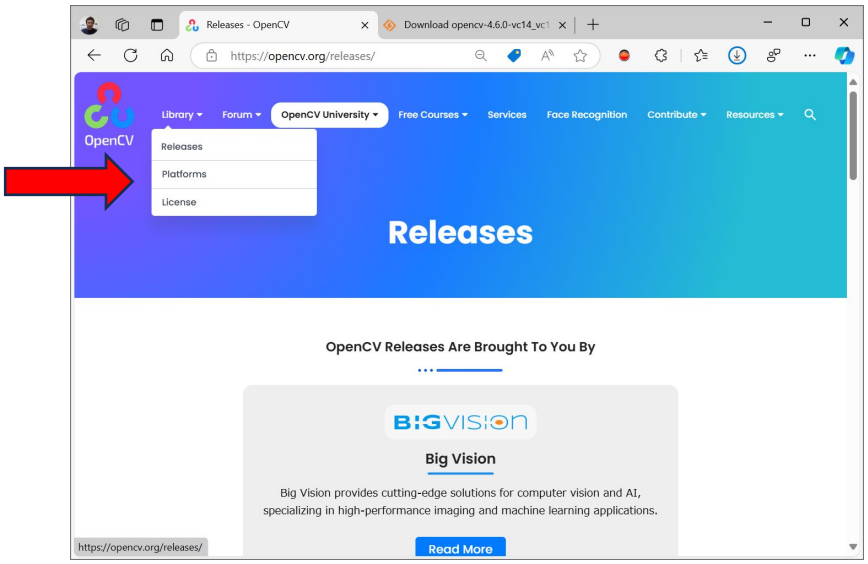
(1) 公式サイト：https://opencv.org/
(2) 日本語リファレンスマニュアルサイト：http://opencv.jp/cookbook/opencv_img.html#

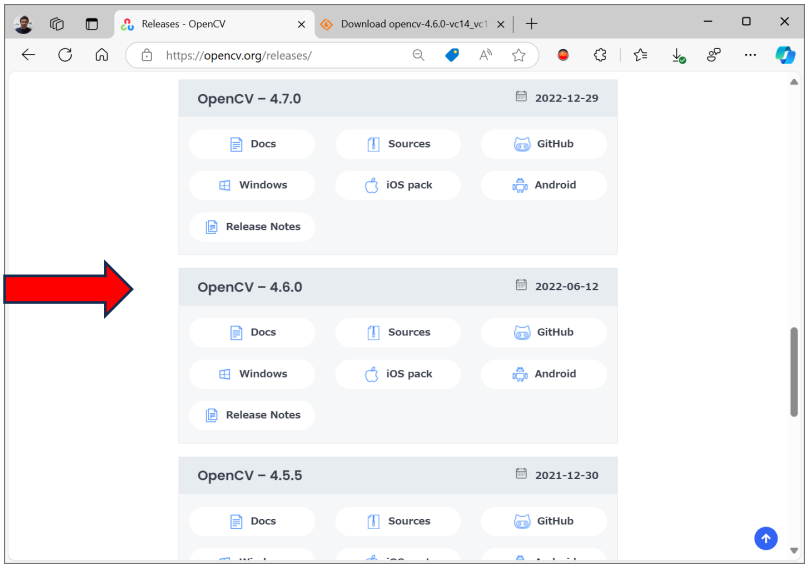
付録 1：VisualStudio2019 における OpenCV 使用環境の設定

① OpenCV ファウルダウロード

OpenCV 公式サイト（https://opencv.org/）から opencv-4.6.0-vc14_vc15.exe をダウンロード

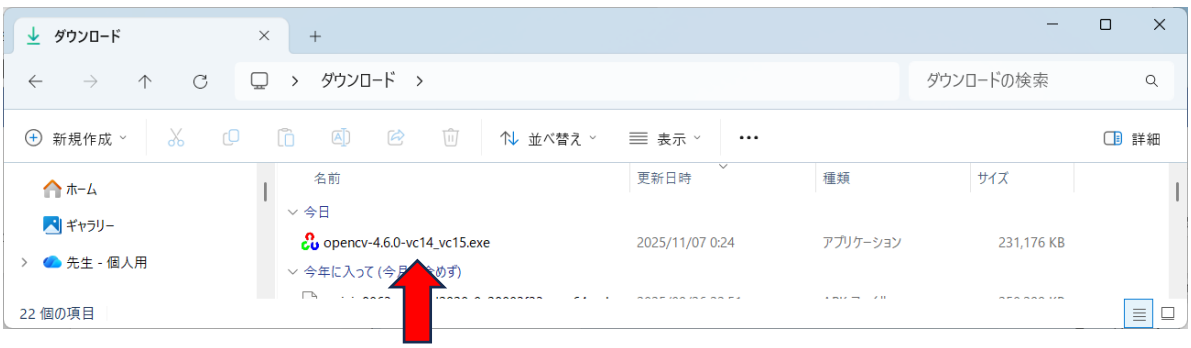
Library → Releases → OpenCV – 4.6.0 → Windows

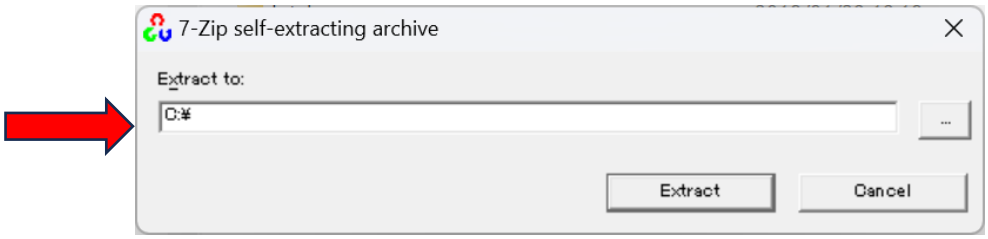


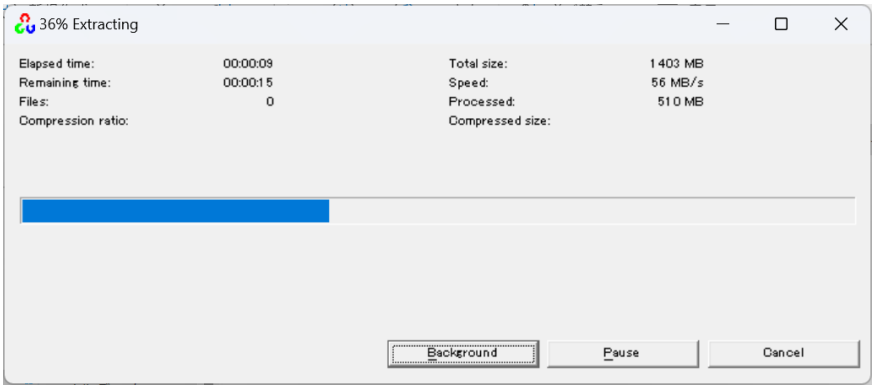


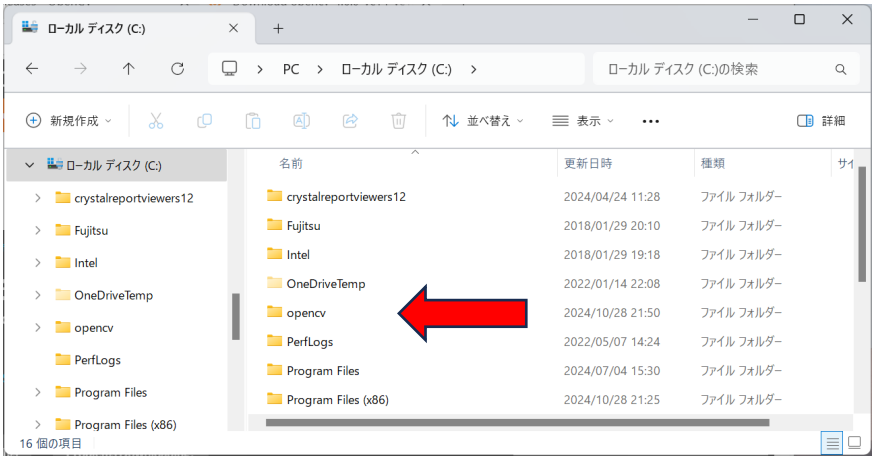
② exe ファイルを展開

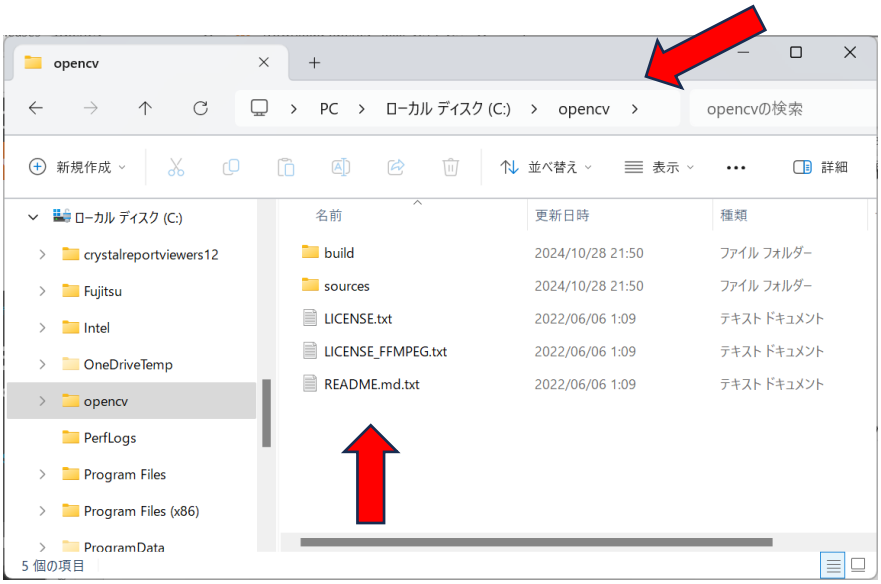
C ドライブの直下に opencv-4.6.0-vc14_vc15.exe を展開する。フォルダ opencv が作成され、そこに opencv のファイルが格納されている。





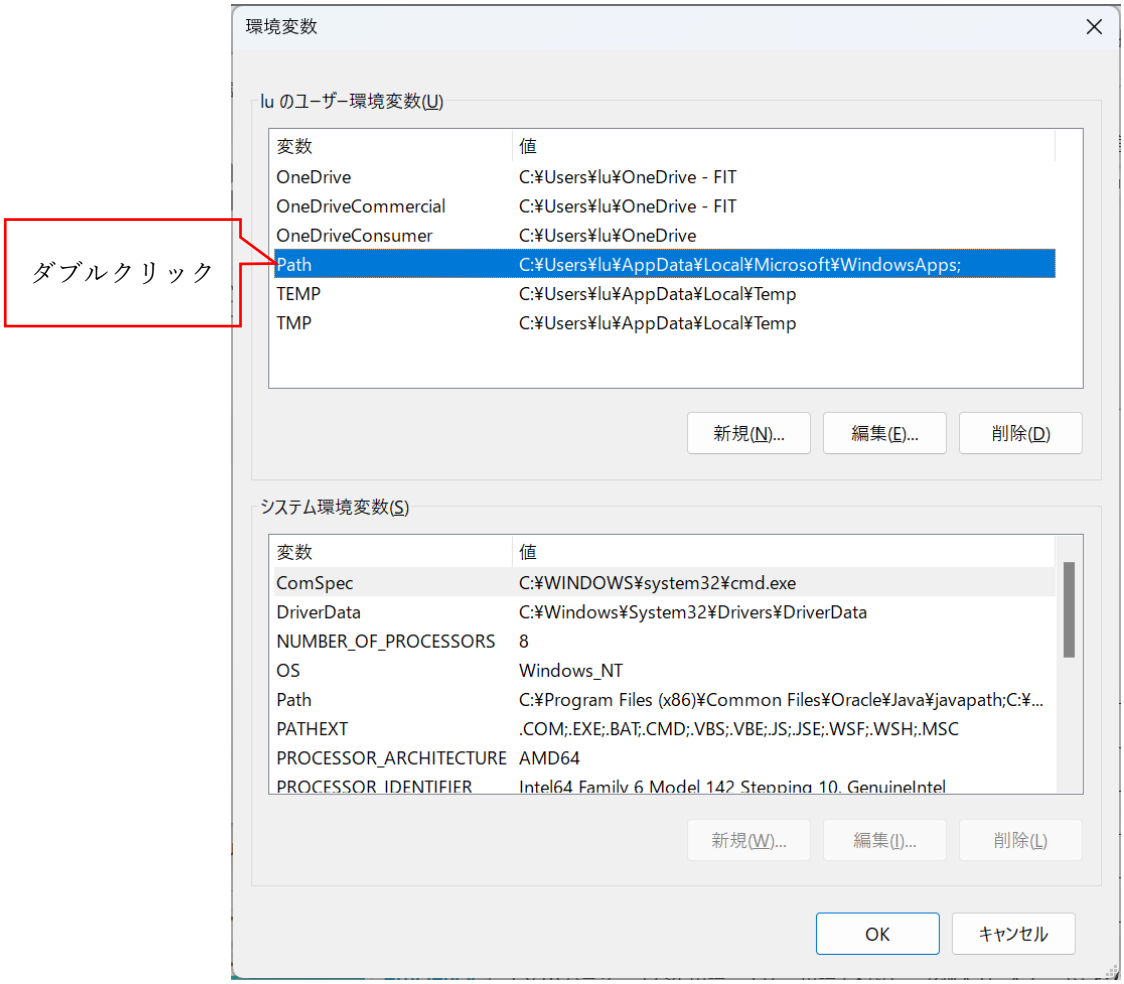




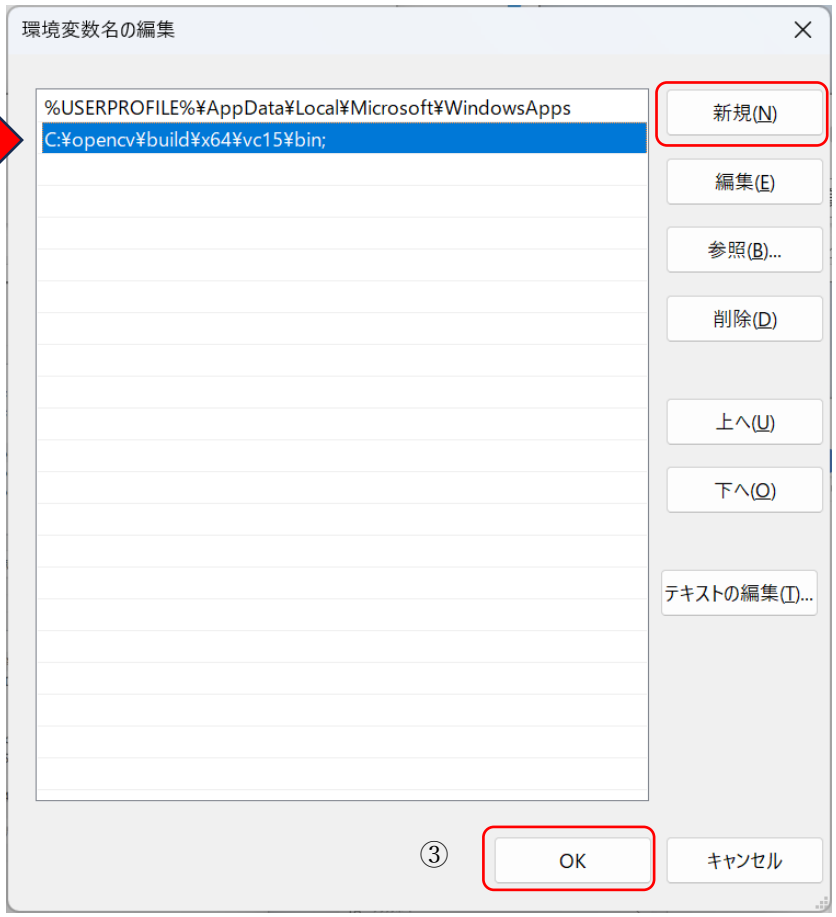


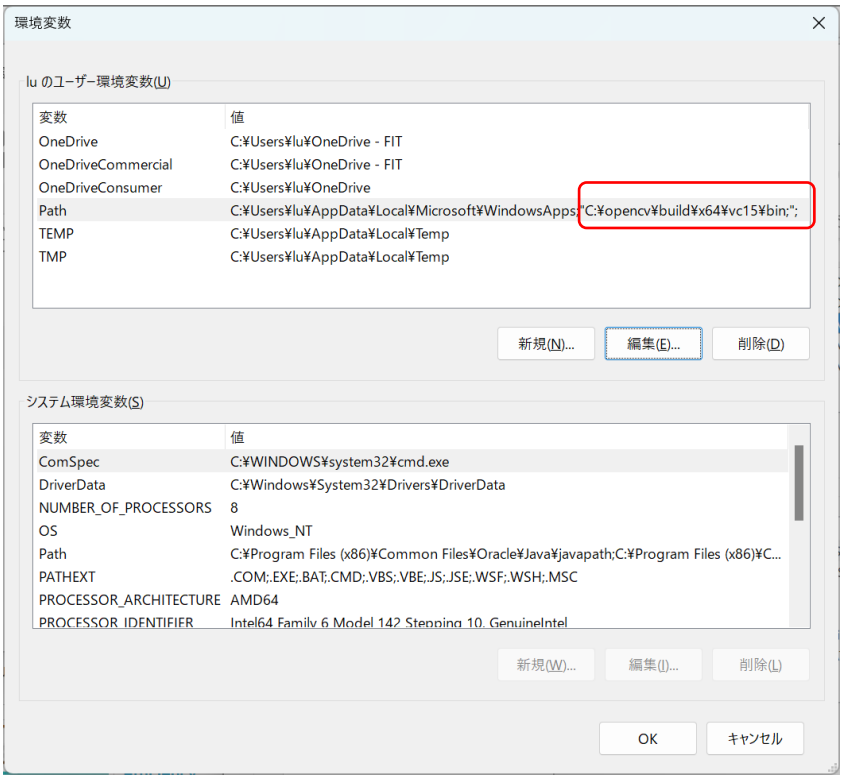
③ 環境変数の「Path」を編集

コントロールパネル→システム→「環境変数を編集」を検索→編集変数編集フォーム Path→新規→「C:¥opencv¥build¥x64¥vc15¥bin;」を追加



新しい環境変数を追加する。





追加され

④ VisualStudio の設定
　前記実験テキストを参照

⑤ opencv フォルダ

```
C: --- opencv --- build --- bin
                          --- etc
                          --- include --- opencv2 --- 今回の実験用各種ファイル
                          --- java
                          --- python
                          --- x64 --- vc14
                                  --- vc15 --- bin
                                           --- lib --- opencv_world460d.lib
                                                       (バージョンにより異なる)
                  --- sources --- ****
                              --- samples ---- ****
                                          --- cpp → サンプルの C ソースファイ
                                          --- data → サンプル画像
```

付録 2：OpenCV の顔認識分類器

OpenCV には以下の Haar-like 特徴分類器があらかじめ用意されています。
C:¥opencv¥sources¥data¥haarcascades

ファイル名	内容
haarcascade_eye.xml	目
haarcascade_eye_tree_eyeglasses.xml	眼鏡
haarcascade_frontalcatface.xml	猫の顔（正面）
haarcascade_frontalcatface_extended.xml	猫の顔（正面）
haarcascade_frontalface_alt.xml	顔（正面）
haarcascade_frontalface_alt2.xml	顔（正面）
haarcascade_frontalface_alt_tree.xml	顔（正面）
haarcascade_frontalface_default.xml	顔（正面）
haarcascade_fullbody.xml	全身
haarcascade_lefteye_2splits.xml	左目
haarcascade_licence_plate_rus_16stages.xml	ロシアのナンバープレート（全体）
haarcascade_lowerbody.xml	下半身
haarcascade_profileface.xml	顔（証明写真）
haarcascade_righteye_2splits.xml	右目
haarcascade_russian_plate_number.xml	ロシアのナンバープレート（数字）
haarcascade_smile.xml	笑顔
haarcascade_upperbody.xml	上半身