WPF/C#による OpenCV 4 プログラミング

リッチなUIと高度な画像処理の融合

北山洋幸◉著

■**サンプルファイルのダウンロードについて**

本書掲載のサンプルファイルは、下記 URL からダウンロードできます。

https://----------

- ファイルサイズを小さくするため、NuGet で導入した OpenCvSharp の packages と、いくつかのバイナリファイルをプロジェクトから削除しています。
- プロジェクトファイルは、そのまま利用できることを保証するものではありません。
- ダウンロードファイルの提供は、ソースコードを入力する手間を省くことを目的としています。

はじめに

OpenCV[※1]は、膨大な関数を用意した画像処理ライブラリ集です。一般的な二次元の画像処理、ヒストグラム処理、ポリゴン処理、テンプレートマッチング、オプティカルフロー、および顔認識など多様なアプリケーションを開発できる関数群を用意しています。これらの関数のリファレンス、ならびに大量のサンプルプログラムは、OpenCV のサイトで紹介されています。また、同サイトには、サンプルコードと共に、それに対する丁寧な説明文まで提供されています。

C# と OpenCV を組み合わせると、スマートなユーザーインタフェースと高度な画像処理を両立できるでしょう。残念ながら C# の本家から OpenCV を利用するインタフェースは提供されていません。しかし、サードパーティから C# と OpenCV を融合させるラッパーがいくつか提供されています。本書では、最も一般的に使われていると思われるラッパーを使用し、C# から OpenCV の機能を使う方法を紹介します。

C# と一口で言っても、Windows フォームアプリ、コンソールアプリ、WPF アプリケーションなどがあり、これらは .NET または .NET Framework と組み合わせることができます。ただ、.NET Framework や Windows フォームアプリはすでに古い技術であり、新規に C# を学習する人にとって、これらを学習する必要性は低いでしょう。このような背景から、本書は WPF アプリケーションと .NET の組み合わせで OpenCvSharp を使う方法を紹介します。WPF を使いますが、デザインパターンに MVVM は採用せず、Windows フォームアプリに慣れた人が学習しやすいコードビハインドで記述します。これは、小さなプログラムではそのほうが見通しがよく、MVVM は冗長と考えたためです。

対象読者

- C# から OpenCV を利用したい人
- OpenCV を WPF で利用したい人
- OpenCV とリッチなユーザーインタフェースが同居したプログラムを開発したい人

謝辞

出版にあたり、開発環境である Visual Studio を無償で提供した米 Microsoft 社、無償の OpenCV を提供した Intel 社、そして OpenCvSharp を提供してくださった Shima 氏、写真を提供していただいた馬場宏氏に深く感謝いたします。また、株式会社カットシステムの石塚勝敏氏にも深く感謝いたします。

2023 年初初夏 新型コロナが収まり日常が戻りつつある都立公園にて　北山洋幸

※ 1　Open Source Computer Vision Library

本書の使用にあたって

開発環境、および、実行環境の説明を行います。

- Windows バージョン
  Windows のバージョンへ依存するとは思えませんが、開発・実行を確認したのは Windows 10 Pro（64 ビット）です。ほとんどのプログラムを Windows 10 Home Edition（64 ビット）でも確認していますが、すべてを確実に確認したのは Windows 10 Pro（64 ビット）です。

- Visual Studio のバージョンとエディション
  無償の Visual Studio Community 2022 を使用します。

- OpenCvSharp のバージョン
  バージョン 4.4.0.20200915 以降を利用します。4.7.0.20230115 などでも確認しましたが、執筆中にバージョン変更があったか厳密な観察は行っていません。なるべくバージョン依存を排除したつもりですが、将来のバージョンで動作することを保証するものではありません。

- URL
  文中や参考資料に URL の記載があります。これは執筆時点のものであり、変更される可能性もあります。リンク先が存在しない場合、キーワードなどから自身で検索してください。

用語

用語の使用に関して説明を行います。

- カタカナ語の長音表記
  「メモリー」や「フォルダー」など、最近は語尾の「ー」を付けるのが一般的になっていますので、なるべく「ー」を付けるようにします。ただ、開発環境やドキュメントなどに従来の用語を使用している場合も多いため、本書では、語尾の「ー」は統一していません。なるべく統一を心掛けましたが、参考資料などでも混在して使用しているため、統一が困難でした。このため、従来の表現と最近の表現が混在しています。

- クラスとオブジェクト
  本来はインスタンス化しているためオブジェクトと表現した方がよい場所でも、クラスと表現する場合があります。これは文脈から読み取ってください。

- ユーザーインタフェース
  ユーザーインタフェースを UI や GUI と省略する場合があります。

- コントロールやクラス
  フォームデザイン時などに使用するコントロール、例えば Image などをコントロールやクラス、あるいは単純に Image と表現し、クラス、コントロールを正確に表現していない場合があります。

- フォームとウィンドウ
  フォームとウィンドウは同じものを指します。主に開発時はフォームと呼び、実行時にウィンドウと呼びますが、例外もあります。

- ソースリストとソースコード
  基本的に同じものを指しますが、ソースリストと表現する場合ソース全体を、ソースコードと表現する場合、ソースの一部を指す場合が多いです。

- オブジェクト
  インスタンスと表現した方が良い場合でも、オブジェクトと表現している場合があります。両方を、厳密に使い分けていませんので、文脈から判断してください。あるいは、物体を指す場合もあります。

- 関数とメソッド
  本来ならメソッドと表現した方が良さそうな場合でも、従来の名残か関数という表現がオリジナルドキュメントで採用されているときがあります。本書も、それにならって関数という表現が残っています。

- 返却値と戻り値
  メソッドや関数の「戻り値」と「返却値」を統一していません。このため、「返す」という表現と「戻す」という表現が混在します。

目次

第1章 はじめてのプログラム

1

C# から OpenCV を使用する基本を説明します。はじめてなので、プロジェクトへ OpenCvSharp を導入する方法も解説します。本書のプログラムは WPF で開発しますが、本章では、コンソールアプリと Windows フォームアプリの例も紹介します。

1.1 OpenCvSharp とは

OpenCvSharp は、shimat 氏が開発した C# 用の OpenCV ラッパーです。筆者が調べた限り、C# から OpenCV を利用するときに最も使用され、機能も充実しているラッパーです。導入も簡単ですので、プログラミング初心者に限らず経験者にもおすすめです。

本ラッパーは、OpenCV で画像保持に使われる `Mat` クラスと .NET の `BitmapSource` 間で相互に変換する機能を用意しています。このため、OpenCV で処理した結果を容易に C# のコントロールへ表示することが可能です。また、本書では執筆時の最新バージョンを使用しますが、OpenCV の各バージョンに対応した NuGet パッケージも用意されています。OpenCvSharp は開発が続いているため、より早く最新の OpenCV を使用できます。この点も OpenCvSharp を推奨する重要な要素です。

画像保持

Windows フォームアプリを利用する場合、OpenCvSharp がサポートする `Mat` を `Bitmap` へ変換する機能を利用します。しかし、WPF アプリケーションでは、`BitmapSource` を利用しますので、画像の保持は Windows フォームアプリと WPF アプリケーションで若干異なります。詳細については、本文で解説します。

1.1.1 ライセンスについて

OpenCvSharp は LGPL ライセンスを採用しています。OpenCV 自体は BSD ライセンスです。ほかにも OpenCV などで使用されるライブラリ、たとえば TBB や ffmpeg などは個別にライセンスが存在します。

商用利用する場合や、本体のソースコードを配布したり、あるいは改変して使用する場合は、ライセンスに違反しないように十分配慮してください。

1.2 はじめてのプログラム

本章では、最も単純と思われる C# から OpenCV を利用するプログラムを紹介します。1.3 節ではコンソールアプリの例、1.4 節では Windows フォームアプリの例、1.5 節では WPF アプリケーションの例を紹介します。なお、プロジェクトへ NuGet で OpenCvSharp を導入する方法も説明します。

1.3 コンソールアプリ

まず、新しいコンソールプロジェクトを作成します。

(1) Visual Studio 2022 Community を起動し、［ファイル］→［新規作成］→［プロジェクト］を選びます。

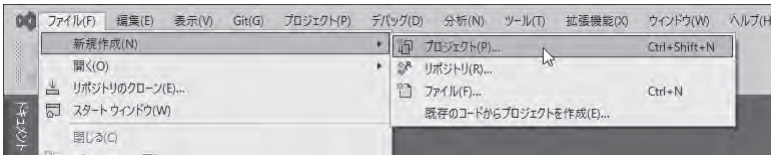

図1.1●［ファイル］→［新規作成］→［プロジェクト］を選ぶ

起動時に Visual Studio 2022 の画面が現れますので「新しいプロジェクトの作成」を選び、プロジェクトを新規に作成しても構いません。

(2)「新しいプロジェクトの作成」画面が現れますので「コンソール アプリ」を選びます。「コンソール アプリ (.NET Framework)」を選んでも構いませんが、.NET Framework は開発が止まりますので、なるべくなら「コンソール アプリ」を選びます。

図1.2●「新しいプロジェクトの作成」画面

(3)「新しいプロジェクトを構成します」画面が現れますので「プロジェクト名」や「場所」などを指定します。

図1.3●「新しいプロジェクトを構成します」画面

(4) これでプロジェクトが作成されます。このプロジェクトには、自動で生成されたソースファイル（Program.cs）が含まれます。

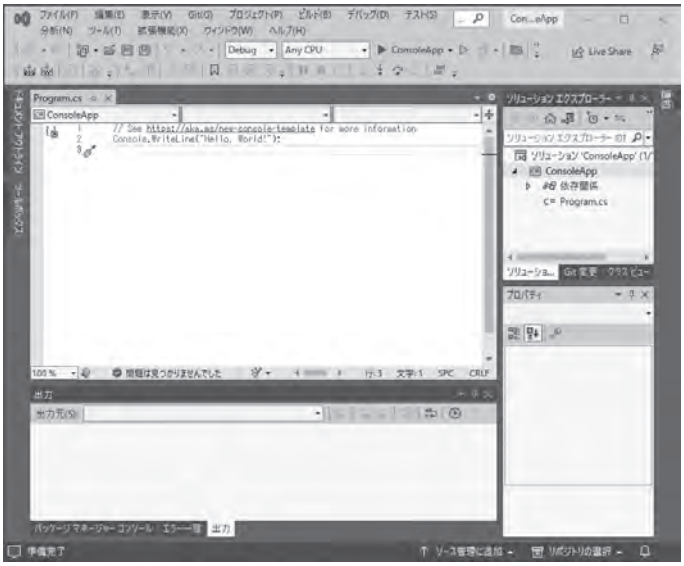

図1.4●プロジェクト完成

リスト1.1●自動で生成されたProgram.cs

```
// See https://aka.ms/new-console-template for more information
Console.WriteLine("Hello, World!");
```

(5) このプログラムをビルドして実行すると、「Hello World!」が表示されます。

(6) このソースファイルを書き換え、OpenCV へ対応させたものを示します。まだ OpenCvSharp を導入していないため、関連するソースコードの部分でエラーが発生し、画面に赤い波線の下線で示されます。

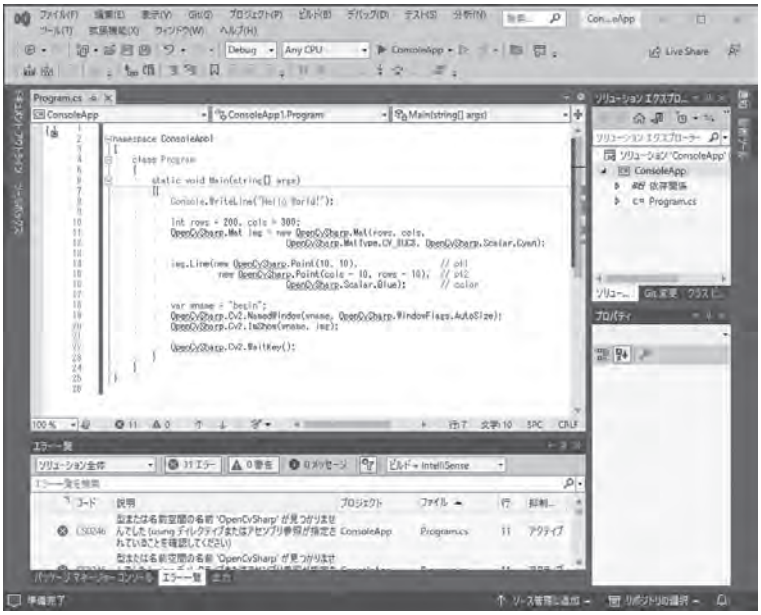

図1.5●ソースファイルを書き換えた状態

1.3.1 NuGet を用いて OpenCvSharp を導入

OpenCvSharp の導入は NuGet を用いると簡単です。NuGet 上に公開されている OpenCvSharp はいくつかのバージョンやプラットフォーム用が存在しますので、間違ったものを導入しないように慎重に選択しましょう。

(1) ソリューションエクスプローラーのプロジェクトの上で、マウスの右ボタンを押し、［NuGet パッケージの管理］を選択します。

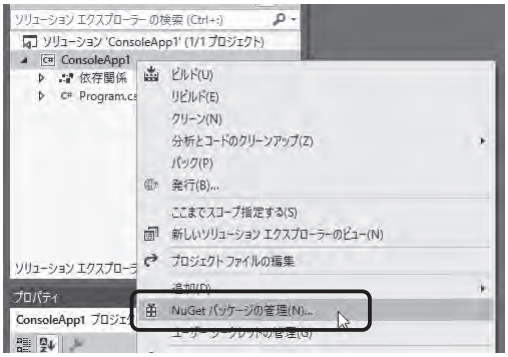

図1.6●［NuGetパッケージの管理］を選択

(2)「NuGet パッケージマネージャー」が現れますので、「参照」を選びます。

図1.7●「参照」を選ぶ

(3) このままパッケージを探しても良いのですが、数が多いので検索欄に「opencvsharp」あるいは「opencvsharp4」（バージョン 4.x の場合）と入力し、OpenCvSharp 関連のパッケージを表示させます。

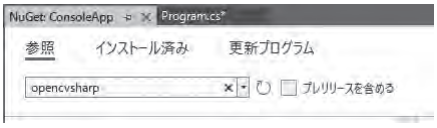

図1.8●検索

(4) 今回は OpenCV のバージョン 4.x を利用したいため、OpenCvSharp4.Windows を選び［インストール］をクリックします。細かなバージョンや公開日、ライセンス情報などが表示されますので、詳細は画面で確認してください。

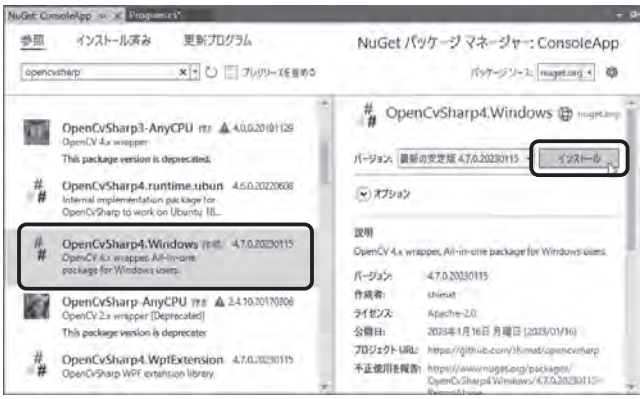

図1.9●OpenCvSharp4.Windowsをインストール

［インストール］をクリックしたときに確認のダイアログが現れることもありますので、適宜応答してください。

(5) しばらくするとインストールが完了します。正常にインストールが完了すると、ソースコードの下部に表示されていた赤い線が消え、OpenCvSharp のパッケージが導入されたのがわかります。

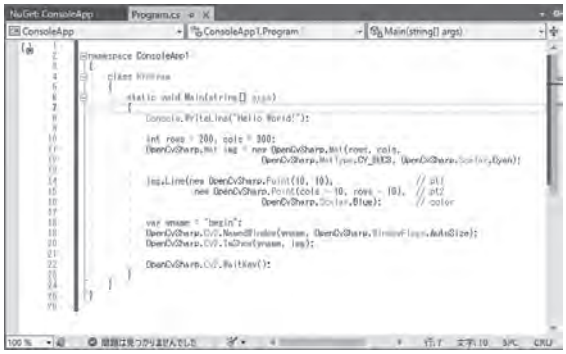

図1.10●インストール完了

1.3.2 実行例

ここまで完了すると、以降は一般的な C# のプログラムと同様です。ビルドして実行した例を示します。プログラムが起動すると、画像が表示されます。

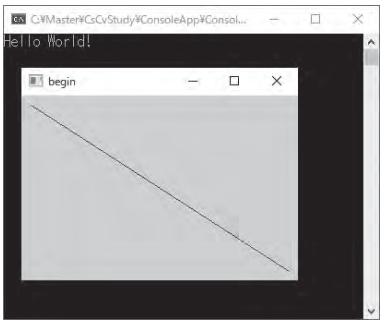

図1.11●実行例

プログラムを終了させたい場合、画像表示ウィンドウにフォーカスを移動してから何かキーを押してください。コンソールウィンドウにフォーカスを置いたままでは、プログラムへキーコードが渡りませんのでプログラムは終了しません。

最終のソースリストを次に示します。

リスト1.2●最終のProgram.cs

```
namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");

            int rows = 200, cols = 300;
            OpenCvSharp.Mat img = new OpenCvSharp.Mat(rows, cols,
                            OpenCvSharp.MatType.CV_8UC3, OpenCvSharp.Scalar.Cyan);

            img.Line(new OpenCvSharp.Point(10, 10),                 // pt1
                     new OpenCvSharp.Point(cols - 10, rows - 10),  // pt2
                                    OpenCvSharp.Scalar.Blue);       // color

            var wname = "begin";
            OpenCvSharp.Cv2.NamedWindow(wname, OpenCvSharp.WindowFlags.AutoSize);
            OpenCvSharp.Cv2.ImShow(wname, img);

            OpenCvSharp.Cv2.WaitKey();
        }
    }
}
```

OpenCvSharp に対する using を追加していないため、OpenCvSharp を利用した部分へは、必ず「OpenCvSharp.」を付加します。

実行結果から分かるように、Mat 生成時に横幅 300 ピクセル、高さ 200 ピクセルで、色がシアンのものを作成します。その後、Line で斜めに青色の線を引きます。

1.3.3　OpenCvSharp4.Extensions や OpenCvSharp4.WpfExtensions の導入

このプログラムでは不要でしたが、ここからさらに拡張したり、あるいは WPF アプリケーションや Windows フォームアプリを開発する際に、「OpenCvSharp4.Extensions」や「OpenCvSharp4.WpfExtensions」が必要になる場合があります。そのような場合は、先の「OpenCvSharp4.Windows」をインストールしたときと同じように、NuGet を利用してインストールしてください。以降に、インストールの様子を示します。

図1.12●OpenCvSharp4.Extensionsをインストール

図1.13●OpenCvSharp4.WpfExtensionsをインストール

このままでは、C++ で開発したプログラムと大して変わりありません。そこで、1.4 節で Windows フォームアプリの例を、1.5 節で WPF アプリケーションの例を紹介します。基本的に本書は、「WPF アプリケーション」プロジェクトで OpenCvSharp を利用します。

1.4 Windows フォームアプリ

ここでは、Windows フォームアプリで OpenCvSharp を使用する例を紹介します。

(1) Visual Studio 2022 Community を起動し、［新規作成］→［プロジェクト］を選びます。
(2)「新しいプロジェクトの作成」画面が現れますので「Windows フォームアプリ」を選びます。
(3)「新しいプロジェクトを構成します」画面が現れますので「プロジェクト名」や「場所」などを指定します。
(4) これでプロジェクトが作成されます。このプロジェクトには、自動で生成されたフォームとソースファイルが含まれます。

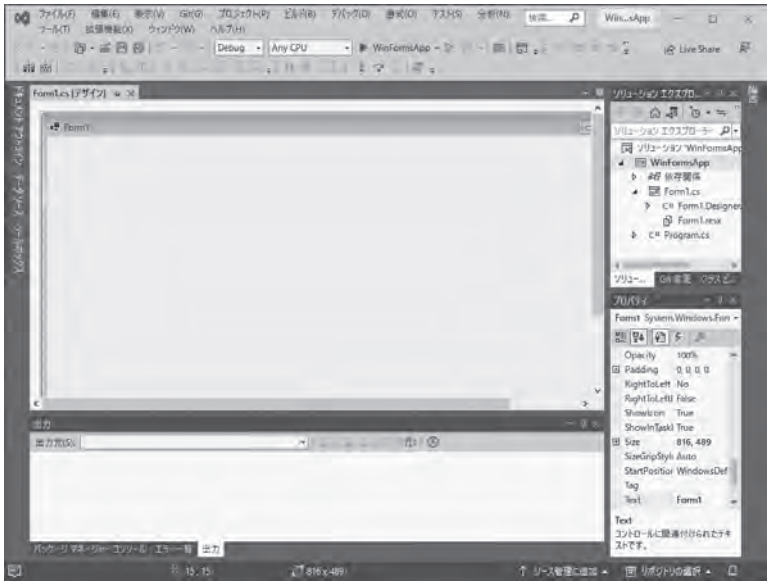

図1.14●プロジェクト完成