C#で始める OpenCV 4 プログラミング

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、一部を除いてインターネット上のダウンロードサービスからダウンロードすることができます。詳しい手順については、本書の巻末にある袋とじの内容をご覧ください。

なお、ダウンロードサービスのご利用にはユーザー登録と袋とじ内に記されている番号が必要です。そのため、本書を中古書店から購入されたり、他者から貸与、譲渡された場合にはサービスをご利用いただけないことがあります。あらかじめご承知おきください。

- ファイルサイズを小さくするため、NuGet で導入した OpenCvSharp の packages をプロジェクトから削除しています。
- プロジェクトなどのファイルは、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。

・本書の内容についてのご意見、ご質問は、お名前、ご連絡先を明記のうえ、小社出版部宛文書（郵送または E-mail）でお送りください。
・電話によるお問い合わせはお受けできません。
・本書の解説範囲を越える内容のご質問や、本書の内容と無関係なご質問にはお答えできません。
・匿名のフリーメールアドレスからのお問い合わせには返信しかねます。

本書で取り上げられているシステム名／製品名は、一般に開発各社の登録商標／商品名です。本書では、™ および ® マークは明記していません。本書に掲載されている団体／商品に対して、その商標権を侵害する意図は一切ありません。本書で紹介している URL や各サイトの内容は変更される場合があります。

はじめに

OpenCV（Open Source Computer Vision Library）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が、丁寧な説明文とともにOpenCVのサイトに紹介されています。

このように便利なライブラリですが、残念なことに、C#の本家から正式なOpenCVを利用するインターフェースは提供されていません。しかしながら、第三者から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。また、C#から単純にOpenCVを使用するだけではなく、C#で構築したリッチなユーザーインターフェース（UI）を活用するために、OpenCVが処理した結果をC#へ取り込む方法も紹介します。

OpenCV自体の解説はたくさんの書籍が発行されており、OpenCvSharpやOpenCVのサイトには、豊富なサンプルプログラムが含まれています。また、これらのサンプルを書き直し、解説を加えたブログも多数存在します。このような背景から、本書は多くのサンプルプログラムを掲載することへ注力するのではなく、C#からOpenCVを使用したい人の参考、あるいはC#のフォームとOpenCVの融合にフォーカスして解説します。とはいえ、サンプルプログラムが、それほど少ないわけではありません。

空間フィルタ処理やヒストグラム、およびアフィン変換などについては、.NET Frameworkにも十分なクラスが用意されているため、C#の機能で比較的容易に開発できます。しかし、もっと高度な機能を実現しようとするとOpenCVに一日の長があります。ところが、OpenCVをCやC++言語で使用すると、UIは貧相なものとなります。このため、せっかく高度な画像処理ができても、見栄えは陳腐なものになりがちです。そこで、本書はC#とOpenCvSharpを使用することによって、リッチなUIと高度な画像処理を融合させます。このような方法を採用すると、高度な画像処理と使いやすいユーザーインターフェース、そして素晴らしいGUIをもつアプリケーションを開発できます。

対象読者

- C# から OpenCV を利用したい人
- 画像処理プログラミング入門者
- OpenCV とリッチなユーザーインター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 を無償公開している米 Microsoft 社、OpenCV を無償公開した米 Intel 社、そして OpenCvSharp を公開されている shimat 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。

2020 年初秋 新型コロナの流行で外出自粛中の自宅にて　北山洋幸

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、本書のプログラムの開発および動作確認は Windows 10 Pro（64 ビット）で行いました。Windows 10 Home Edition（64 ビット）でもほとんどのプログラムの動作を確認しています。

■ Visual C# のバージョンとエディション

無償の Visual Studio Community 2019 を使用します。

■ OpenCvSharp のバージョン

バージョン 4.4.0.20200915 を使用しました。

■ URL

URL の記載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分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く、参考資料も混在して使用しているため、本書では、語尾の「ー」は統一していません。

■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。

■ユーザーインターフェース

ユーザーインターフェースを UI や GUI と省略する場合があります。

■コントロール

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PictureBoxコントロールを、PictureBoxと表現し、コントロールを省く場合があります。

■フォームとウィンドウ

GUIをデザイン時にフォームと呼び、実行時にウィンドウと呼ぶことがあります。

■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する場合があります。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。

■映像とフレーム

カメラから取得した画像を映像と表現する場合とフレームと表現する場合が混在しますが、同じものを指します。これらは文脈から判断してください。

■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、それに倣って関数という表現を使用する場合があります。

■動画、画像、フレーム

これらは混在して使用しています。動画は画像の集合です。ある瞬間では、動画も画像です。このため、動画を画像と表現した方が適切な場合があります。また、動画はフレームに分割できますので、フレームと表現する場合もあります。画像とフレームは、ほぼ同様の意味ですが、文章の流れからフレームと表現した方が良い場合、フレームを採用します。

目次

はじめに iii

第1章 はじめてのプログラム 1

1.1 OpenCvSharp とは 1
ライセンスについて 2
1.2 コンソールアプリの作成 2
コンソールプロジェクト作成 2
NuGet を用いて OpenCvSharp を導入 6
実行例 8
1.3 Windows フォームアプリケーションの作成 10
実行例 14
1.4 OpenCvSharp のクラスやメソッド 15
Mat クラス 15
Cv2.ImShow 15
Cv2.WaitKey 15

第2章 色や輝度の処理 17

2.1 画像の反転 17
Cv2.ImRead 22
Cv2.BitwiseNot 23
2.2 グレイスケール 24
Cv2.CvtColor 25
2.3 輝度平滑化 25
Cv2.EqualizeHist 27
2.4 閾値処理（スレッショルド処理） 27
Cv2.Threshold 28

第3章 フィルタ処理 31

3.1 ブラー処理 31
CIo クラス 35
Cv2.Blur 37

3.2 ガウシアン処理 38
Cv2.GaussianBlur 39
3.3 ラプラシアン処理 40
Cv2.Laplacian 40
3.4 Sobel 処理 41
Cv2.Sobel 42
3.5 Canny 処理 43
Cv2.Canny 44
3.6 画像の膨張 44
Cv2.Dilate 45
3.7 画像の収縮 46
Cv2.Erode 47
InputArray や OutputArray について 48
3.8 2 画面 48
Cv2.ImWrite 55
3.9 ドラッグ & ドロップ 56
3.10 ユーザーインターフェースと OpenCvSharp を分離 59
CCv クラス 64
CCv クラスの private メソッド 64
CCv クラスの public メソッド 65
3.11 ガンマ補正 67
Cv2.LUT 69

第 4 章 アフィン変換 71

4.1 フリップ 71
Cv2.Flip 73
4.2 リサイズ 74
Cv2.Resize 75
4.3 回転 76
Cv2.GetRotationMatrix2D 77
Cv2.WarpAffine 78
4.4 透視投影 79
Cv2.GetPerspectiveTransform 81
Cv2.WarpPerspective 81

第５章　画像合成 83

5.1　２つの画像を加算 83
Cv2.Add 90
5.2　２つの画像の差分 91
Cv2.Absdiff 92
5.3　２つの画像の論理和 92
Cv2.BitwiseOr 93
5.4　マスクを使った２つの画像加算 94
5.5　ROI を設定した２つの画像加算 96
5.6　重みを付けて２つの画像加算 98

第６章　オブジェクト 103

6.1　コーナー検出（1） 103
Cv2.GoodFeaturesToTrack 105
6.2　コーナー検出（2） 106
Cv2.FindContours 108
6.3　矩形の検出 109
Cv2.DrawContours 114
Cv2.ApproxPolyDP 115
6.4　ノイズの除去 116
Cv2.Inpaint 117
6.5　透視投影 118
6.6　細線化 130
CvXImgProc.Thinning 131

第７章　マウスでオブジェクト操作 133

7.1　透視投影 133
7.2　オブジェクト除去 143
7.3　オブジェクトのサイズを変更 153
Form1.cs の変更点 154
Form2.cs の変更点 154

7.4 オブジェクトのサイズを変更・ガウス関数 161
Form1.cs の変更点 161
Form2.cs の変更点 162
Cv2.Remap 168

第 8 章 オブジェクトの検出とサイズ変更 169

8.1 オブジェクト検出 169
CascadeClassifier クラスのコンストラクター 174
CascadeClassifier.DetectMultiScale 175
8.2 自動でオブジェクトを検出し拡大 176
8.3 自動でオブジェクトを検出し拡大・ガウス関数 178
8.4 テンプレートマッチング 179
Cv2.MatchTemplate 182
Mat.MinMaxLoc 183
8.5 オブジェクト交換 184

第 9 章 特徴点 191

9.1 特徴点検出 191
Feature2D.DetectAndCompute 193
Cv2.DrawKeypoints 193
9.2 特徴点のマッチング 194
9.3 特徴点のマッチング・双方向 197
9.4 パノラマ 200
Stitcher クラス 201
Stitcher.Create メソッド 202
Stitcher.Stitch メソッド 202
9.5 パノラマ・非同期 202

第 10 章 フーリエ変換 209

10.1 離散フーリエ変換 209
Cv2.GetOptimalDFTSize 212
Cv2.CopyMakeBorder 212
Mat.ConvertTo 213
Cv2.Merge 213
Cv2.Dft 214

Cv2.Split 216
Cv2.Magnitude 216
Cv2.Log 217
Cv2.Normalize 217
10.2 逆変換 221
Cv2.Idft 224

第 11 章 動画 225

11.1 カメラ表示・コンソール 225
VideoCapture クラス 227
VideoCapture コンストラクター 227
VideoCapture.Read 228
Window.ShowImage 228
11.2 動画ファイル表示・コンソール 228
11.3 フォームへ動画表示 230
11.4 動画へ画像処理 237

付 録 Visual Studio のインストール 239

参考文献 244

索引 245

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。初めてなので、プロジェクトへ OpenCvSharp を導入する方法も解説します。コンソールアプリケーションを使用し、C++ などで開発したプログラムと同様の UI を持つもの、そして、C# らしく GUI を持つプログラムの基礎を紹介します。

1.1 OpenCvSharp とは

OpenCvSharp は、shimat 氏が開発した C# 用の OpenCV ラッパーです。筆者が調べた限りでは、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。

本ラッパーには、OpenCV で画像保持に使われる `Mat` クラスと .NET Framework の `Bitmap` 間で相互に変換する機能も用意されているため、容易に OpenCV で処理した結果を C# のコントロールへ表示することが可能です。各バージョンに対応した NuGet パッケージが用意されていますので従来の 2.x を使用できますが、本書では執筆時の最新バージョンを使用します。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。

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。

商用に利用する場合や、本体のソースコード配布、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 コンソールアプリの作成

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。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。

コンソールプロジェクト作成

まず、新しいコンソールプロジェクトを作成します。

①Visual Studio 2019 Communityを起動し、[ファイル▶新規作成▶プロジェクト]を選びます。

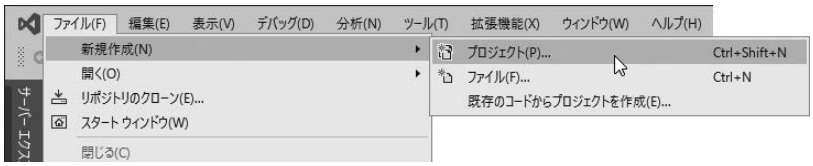

図1.1●［ファイル▶新規作成▶プロジェクト］を選ぶ

起動時に Visual Studio 2019 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。.NET Core を選んでいますが、.NET Framework でも構いません。

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。この例ではデフォルトを使用します。

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。

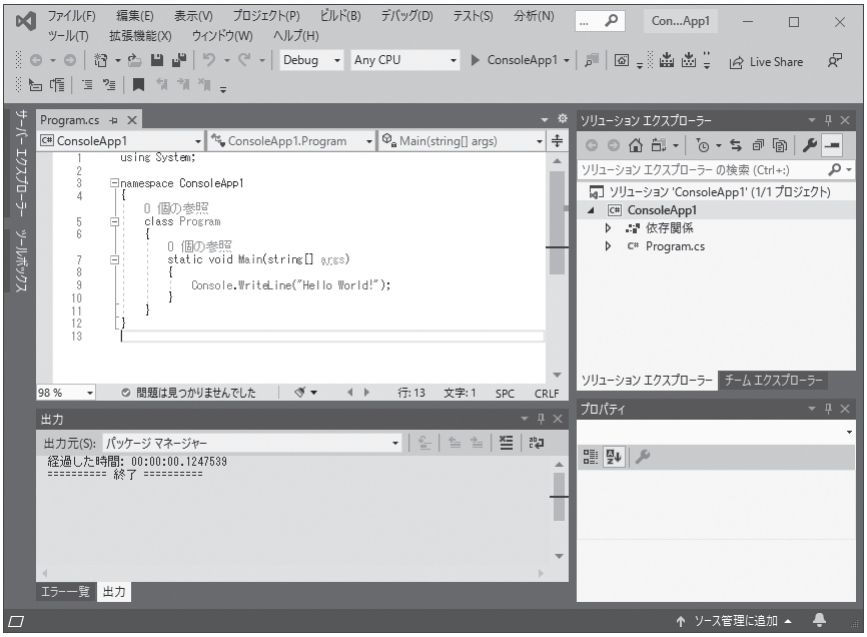

図1.4●プロジェクト完成

リスト1.1●自動で生成されたソースファイル

```
using System;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            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。OpenCvSharp を導入していないため、ソースコードの一部に赤い波線の下線が引かれパッケージが導入されていないのがわかります。

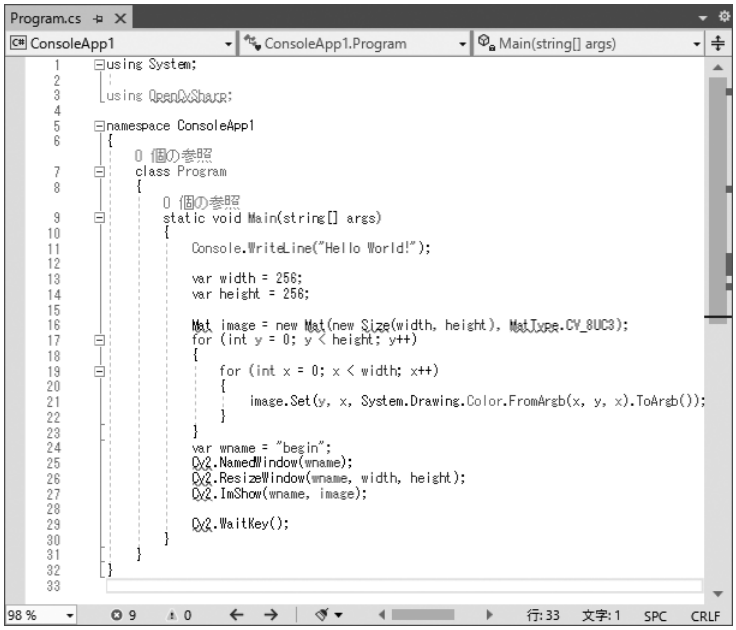

図1.5●ソースファイルを書き換える

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。

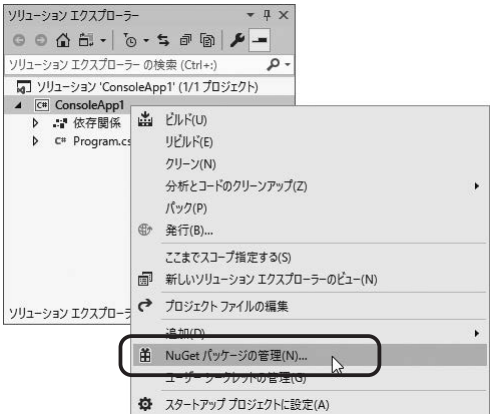

図1.6●［NuGetパッケージの管理］を選択

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。

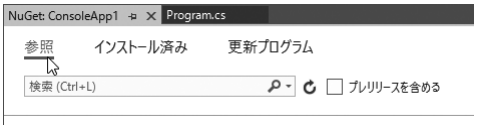

図1.7●「参照」を選ぶ

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」と入力し、OpenCvSharp 関連のパッケージを表示させます。

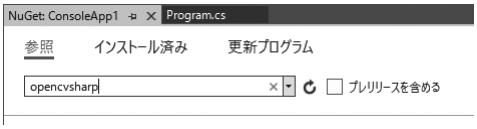

図1.8●検索

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。

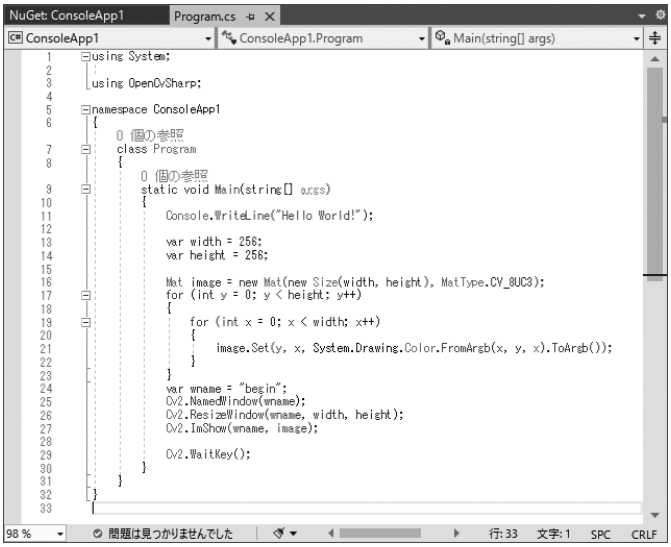

図1.10●インストール完了

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。

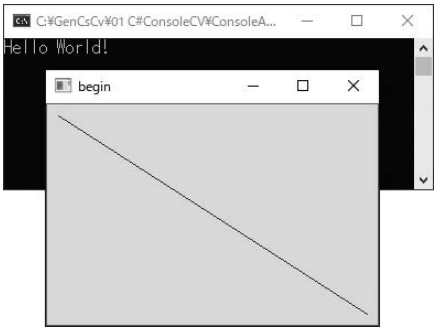

図1.11●プログラムが起動した様子

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以降に示します。

リスト1.2●最終のソースリスト（Program.cs）

```
using System;

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.WindowMode.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。

このままでは、C++ で開発したプログラムと変わりありません。そこで、以降に Windows フォームで開発したプログラムを示します。

1.3 Windows フォームアプリケーションの作成

今度は、Windows フォームアプリケーションで OpenCvSharp を使用する例を紹介します。

① Visual Studio 2019 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 フォームアプリケーション (.NET Framework)」を選びます。

図1.12●「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。この例ではデフォルトを使用します。

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。

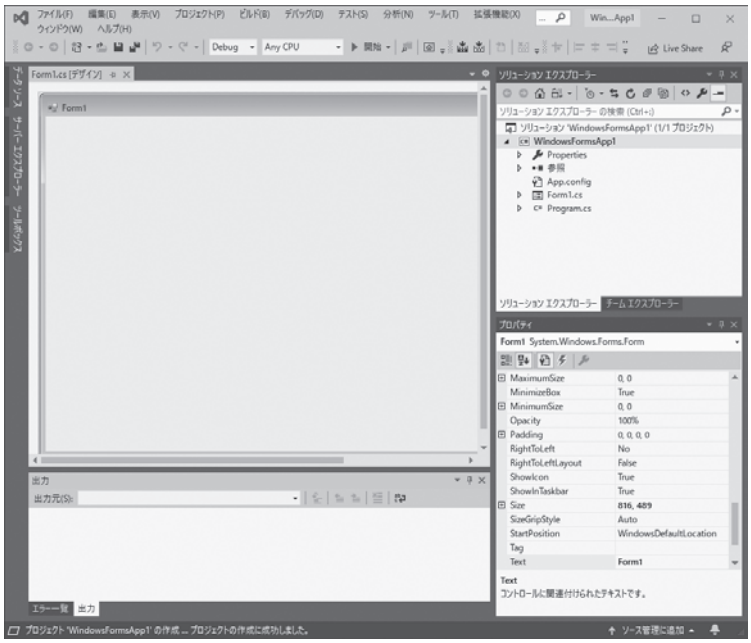

図1.13●プロジェクト完成

自動生成された Form1.cs を示します。

リスト1.3●自動生成されたForm1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

```
    }
}
```

⑤ NuGet を用いて OpenCvSharp を導入する方法は、先に説明した通りです。

⑥フォームの Paint イベントに対応するメソッドを定義します。プロパティのイベントを選択し、Paint 欄をダブルクリックします。すると、Paint イベントに対応したメソッドが定義され、そのソースファイルの部分へカーソルが表示されます。

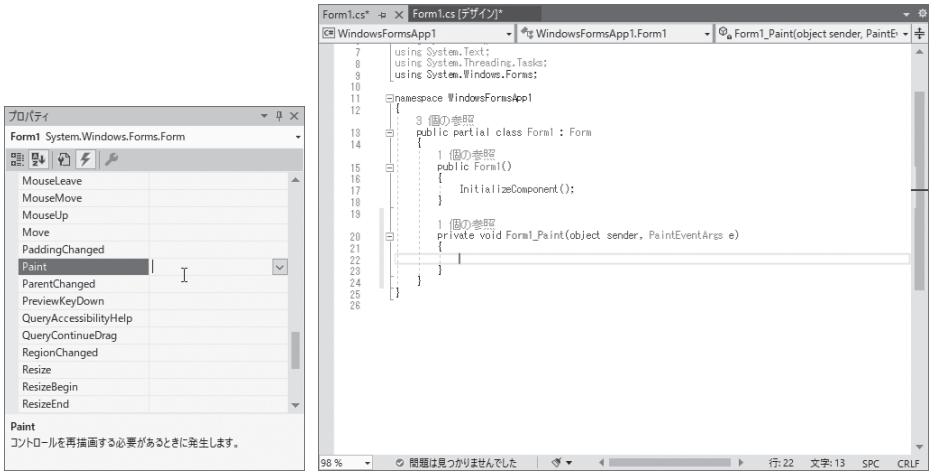

図1.14●Paintイベントに対応するメソッドを定義

⑦ Paint イベントに対応するメソッドを記述します。

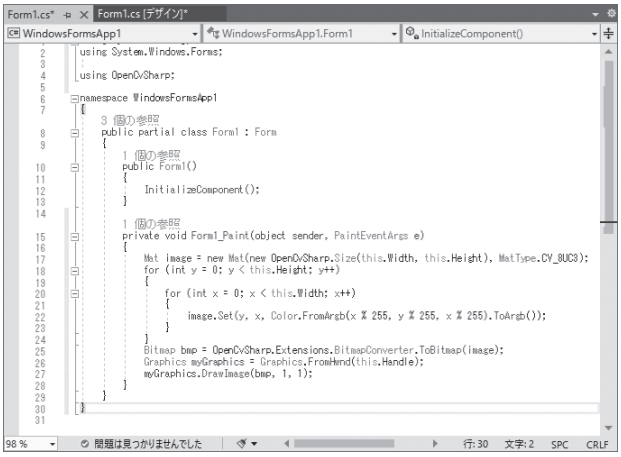

図1.15●Paintイベントに対応するメソッドを記述