广东工程职业技术学院

机器视觉应用软件开发项目案例教程
（c# OpencvSharp 版）

编著：张卿

二零二五年三月

目录

项目 1 机器视觉用户登录程序编写

项目知识目标

- 了解.NET Framework 及 C#语言
- 了解 Visual Studio2015 集成开发环境的安装
- 了解 Visual Studio2015 集成开发环境的基本使用方法
- C# Windows 窗体应用程序的创建步骤
- 掌握窗体、标签、文本框、按钮等控件的基本属性、方法和事件

项目能力目标

- 能够应用 Visual Studio2015 的集成开发环境开发一个简单的 c#程序
- 能够熟练使用窗体、标签、文本框、按钮等控件

任务 1.1 熟悉 Visual Studio2015 的编程环境

学习通扫一扫 看视频

1.1.1.NET Framework(框架)概述

.NET Framework 是一个集成在 Windows 中的组件，它支持生成和运行下一代应用程序与 XML Web Services。.NET Framework 旨在实现一下目标：

提供一个面向对象的编程环境，无论对象代码在本地存储和执行，还是在本地执行但在 Internet 上分布，或者在远程执行。

提供一个将软件部署和版本控制冲突最小化的代码执行环境。

提供一个可提高代码执行安全性的代码执行环境。

提供一个可以消除脚本环境和解释环境的性能问题的代码执行环境。

使开发人员的经验在面对类型大不相同的应用程序（如基于 Windows 的应

用程序和基于 Web 的应用程序）时保持一致。

按照工业标准生成所有通信，以确保基于 .NET Framework 的代码可与任何其他代码集成。

.NET Framework 是 Windows 的托管执行环境，可为其运行的应用提供各种服务。它包括两个主要组件：公共语言运行时 (CLR)，它是处理运行应用的执行引擎；.NET Framework 类库，它提供开发人员可从其自己的应用中调用的已测试、可重用代码库。.NET Framework 提供的用于运行应用的服务包括：内存管理，常规类型系统，一个全面的类库，开发框架和技术，语言互操作性，版本兼容性，并行执行，多定向。

1.1.2 visual c#介绍

C#（读作 C sharp）是一种面向对象的编程语言，是.NET 开发环境的重要组成部分。而 Microsoft Visual C# 2015 是微软开发的 C#编程集成开发环境（同种产品还有 Borland 公司的 C# Builder），它是为生成在 .NET Framework 上运行的多种应用程序而设计的。C# 简单、功能强大、类型安全，而且是面向对象的。C# 凭借它的许多创新，在保持 C 样式语言的表示形式和优美的同时，实现了应用程序的快速开发。

Visual C#（C#）是一种通用、面向对象的编程语言，由微软公司开发。它具有以下主要特点：

简洁易用：C#的语法结构简洁清晰，易于学习和理解。它提供了丰富的编程特性和现代化的语言功能，如自动垃圾回收、类型推断、Lambda 表达式等，使开发人员能够更高效地编写代码。

面向对象：C#完全支持封装、继承和多态等面向对象的概念和技术。开发人员可以使用类、接口、继承和多态等特性来组织和管理代码，提高代码的可维护性和重用性。

强大的.NET 框架：C#是.NET 平台的一部分，可以与.NET 框架紧密集成。通过.NET 框架，开发人员可以使用丰富的类库和组件来实现各种功能，如文件操作、网络通信、数据库访问、图形界面等。

跨平台开发：随着.NET Core 的发布，C#也可以用于跨平台开发。开发人员可以使用 C#编写跨平台的应用程序，包括在 Windows、Linux 和 macOS 等操作系统上运行的应用程序。

强类型检查：C#是一种强类型语言，对变量和数据类型有严格的检查和限制。这有助于减少编程错误和类型相关的问题，提高代码的可靠性和稳定性。

多线程支持：C#提供了多线程编程的支持，开发人员可以使用线程、任务和并行编程库来实现并发和异步操作，从而提高程序的性能和响应性。

与 Web 紧密结合：C#支持 HTML、XML、SOAP 等 Web 标准，便于开发 Web 应用。

安全性强：C#具备强大的安全机制，有助于消除软件开发中的常见错误。垃圾回收器也能有效管理内存资源。

良好的兼容性：C#遵循公共语言规范（CLS），可与其他语言开发的组件兼容。

灵活的版本处理：C#内置了版本控制功能，便于开发人员管理和维护代码。

Visual Studio 包含 Visual C#，这是通过功能齐全的代码编辑器、项目模板、设计器、代码向导、功能强大且易于使用的调试器以及其他工具实现的。通过 .NET Framework 类库，可以访问多种操作系统服务和其他有用的精心设计的类，这些类可显著加快开发周期。

1.1.3 Visual studio2015 的安装

VS2015 共有三个版本，分别是社区版、专业版、企业版。其中社区版免费提供给单个开发人员、 开放源代码项目、科研、教育以及小型专业团队。

Visual studio2015 安装过程。建议在官方网站下载 Visual studio 2015 任意一种版本，本教程采用的社区版。下载完成后会得到一个镜像文件（.ISO 文件），双击该文件即可开始安装。

（1）双击镜像文件后会弹出如下的对话框，选择”运行 vs_community.exe”即可进入安装程序，见图 1。

安装过程中，需要在等待一段时间，根据不同电脑的配置时间长短略有不同。

图 1-1 运行安装程序选择对话框

图 1-2 初始化安装程序

初始化完成后，需要选择安装的类型（自定义或默认值），我们选择定义安装即可。

图 1-3

图 1-4

看到以下图标，就表示安装完成。

图 1-5 安装完成页面

1.1.4 Visual studio2015 的配置

首次使用 VS2015 还需要简单的配置，主要包括开发环境和主题风格。

（1）启动 VS2015 会提示登录，如果你不希望登录，可以点击“以后再说”。

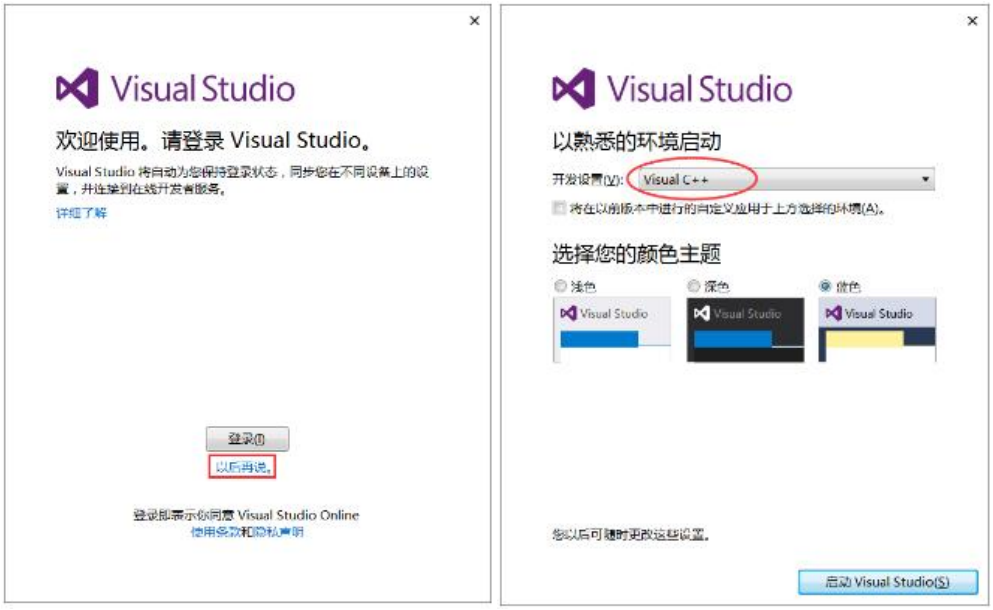


图 1-6

接下来选择环境配置。我们将使用 VS2015 进行 C/C++ 程序开发，所以选择“Visual C++”这个选项。

2）等待几分钟的准备过程，VS2015 就启动成功了。

图 1-7

1.1.5 熟悉 Visual studio2015 c#的编程环境

Visual studio2015 集成开发环境有标题栏、菜单栏、工具栏、工具箱、项目设计区、浮动面板区组成，如图 所示：

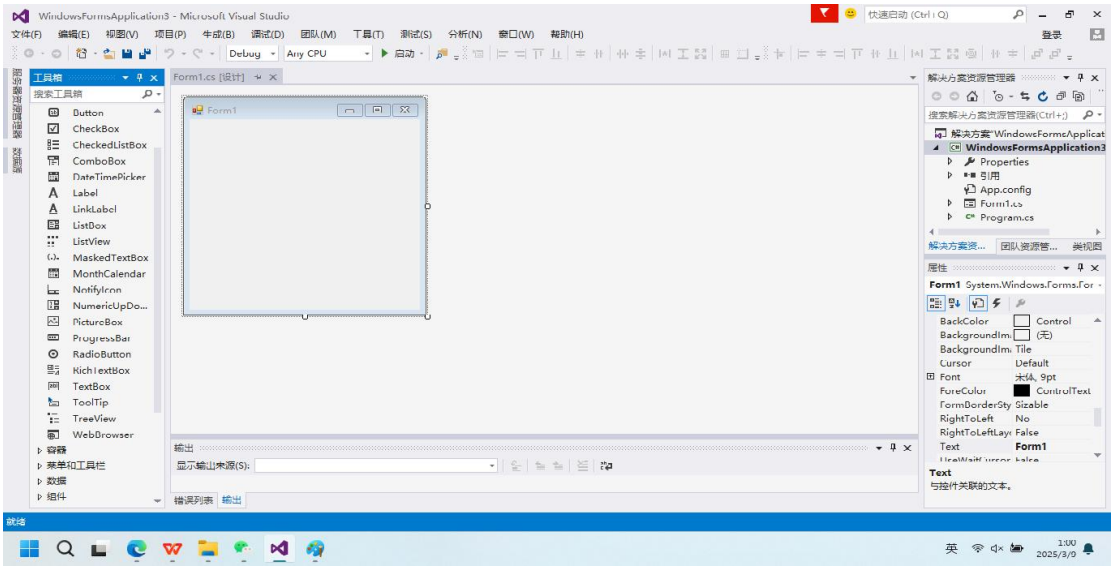


图 1-8

1. 标题栏

标题栏位于窗口的最上方，其作用和其他 windows 窗口基本一眼不过。标题栏用于显示项目的名称以及当前程序所处的状态。

2. 菜单栏

菜单栏中的菜单命令几乎包括了所有的常用功能，包括“文件”、“编辑”、“试图”、“项目”、“工具”、“调试”、“测试”、“分析”、“帮助”等。其中，比较常用的“文件”菜单主要用来新建、打开、保存、关闭项目，“编辑”菜单主要用来剪切、复制、粘贴、删除和查找程序代码。

3、工具栏

工具栏提供了常用的功能按钮。开发人员熟悉工具栏可以大大节省工作时间，提高工作效率。一般工具栏上面有“标准”工具栏和“布局”工具栏。“标准”工具栏讲常用的操作命令以按钮的形式展现，“布局”工具栏将常见的“格式”菜单命令以按钮的形式展现。

4、工具箱

“工具箱”是 visual studio2015 的重要工具，他提供了开发 windows 应用程序所必须的控件。“工具箱”是一个浮动的树控件，与 windows 资源管理器的工作方式非常相似。同时展开“工具箱”的多个段，整个目录树在“工具箱”窗口内部滚动。单机名称旁边的加号“+”，展开“工具箱”的选项卡；单机，名称旁边的减号“—”，折叠一个已展开的选项卡，如下图所示。

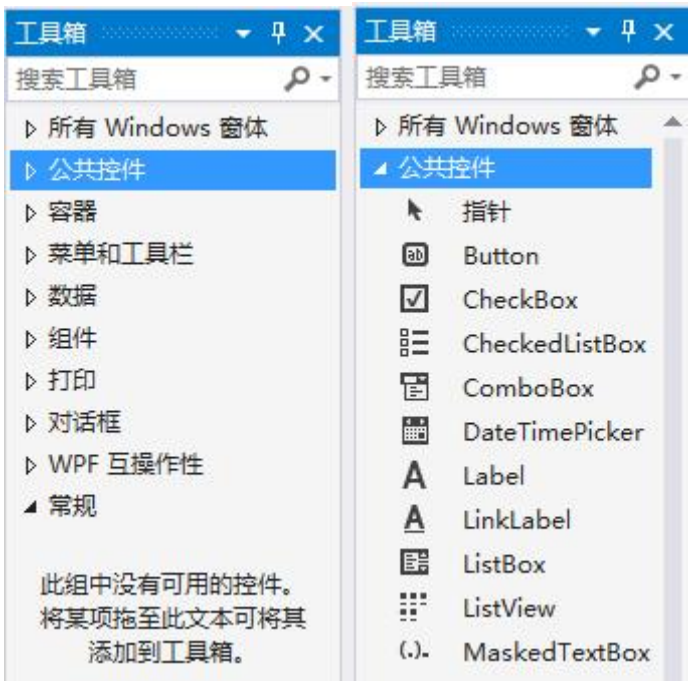
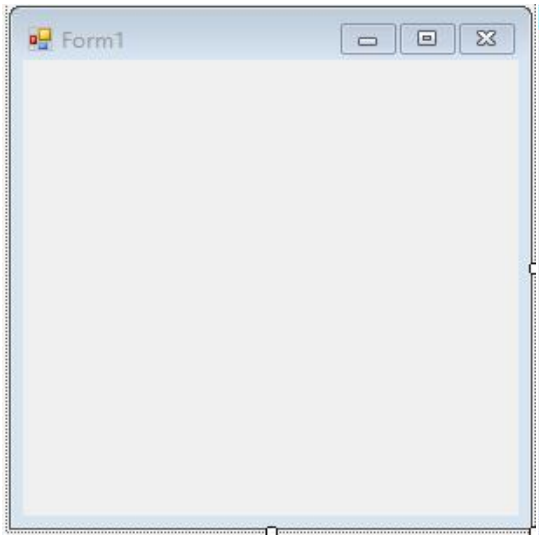


图 1-9 “工具箱”窗体

5、窗体设计器和“代码”窗口



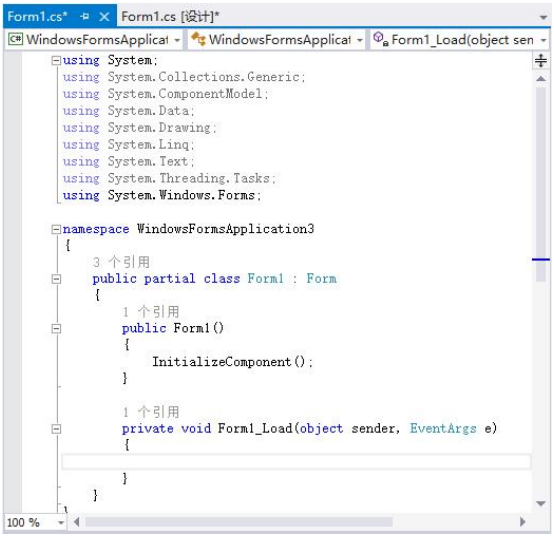


图 1-10 窗体设计器　　图 1-11 “代码窗口”

应用程序设计器为应用程序开发提供一个设计器的界面。其中，窗体设计器用于设置程序的图形用户界面，在“代码”窗口中可以编写代码，如图 ，和图 所示。

6. 解决方案资源管理器

解决方案资源管理器的主要用于管理解决方案或项目。利用解决方案资源管理器，可以查看项目并执行项目管理任务，还可以在解决方案或项目上下文的外部处理文件。

解决方案资源管理器利用树状试图，如图所示，提供项目及其文件的住址关系，并且提供对项目和文件相比命令的便捷访问。从该试图中可以直接打开项目进行修改和执行其他管理任务。由于不同项目的存储项的方式不同，解决方案资源管理器中的文件夹结构不一定反映出所列项的物理存储。与此窗口管理的工具栏提供适用于列表中突出显示项的常用命令。

任务 1.2 Visual studio2015 机器视觉用户登录程序编写

学习通扫一扫 看视频

1.2.1 Windows 窗体应用程序设计流程

在 visual studio 2015 编程环境下开发 visual c#程序一般有一下几个步骤。

1.需求分析

根据实际应用需要，进行需求分析，确定需要设计程序有什么功能，对应功能需要什么控件来实现，以及需要编写什么代码。

2.新建 Windows 窗体应用程序项目

打开 visual studio 2015，新建一个 visual c#应用程序。一个应用程序就是一个项目，或者叫做解决方案，用户根据所需要创建的程序要求，选择合适的应用程序类型。

3.布局程序界面

建立项目后，根据程序的功能要求，在窗体上合理地布置控件，并调整到合适的大小和位置。

4.设置对象的属性

布局好控件后，要设置控件的外观以及初始状态，以满足程序的需要。可以打开属性窗口设置属性。

5.编写代码

布局好控件并设置好控件的初始属性后，就可以编写代码。邮寄控件或窗体，通过“属性”窗口选择需要编写的事件，也可以直接进入“代码”窗口编写代码。

6.运行调试程序

完成上述步骤后，就可以运行程序，并做测试，以便发现问题及时修改。调试和改错是程序开发过程中非常重要的步骤，需要反复使用，以尽量优化程序。

7.生成可执行文件

程序开发完成并正确运行后，需要将其生成课执行文件发布出去。

8.部署应用程序

编写好的应用程序可以在 visual studio 2015 中部署，以自动创建安装文件。

1.2.2 设计机器视觉软件用户登录程序界面设计

1. 要求

设计一个机器视觉软件用户登录界面，对用户输入的用户名和密码进行验证。假设正确的用户名为”admin”，密码为”user”。如果用户名和密码验证成功，讲进入登录成功后的界面。用户登录界面如图 1-12 所示，用户登录成功后的界面如图 1-13 所示。

图 1-12 用户登录界面

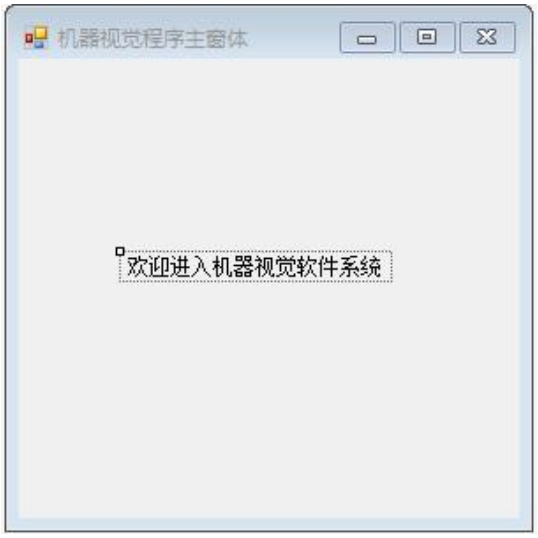


图 1-13 登录成功后的界面

2、设计步骤

（1）新建项目。启动 Visual studio 2015，在“文件”菜单下，选择“新建”菜单下级的“项目”命令，在弹出的“新建项目”对话框选择“windows 窗体应用程序”，然后设置项目的名称 和保存的路径。如图 1-14 所示：

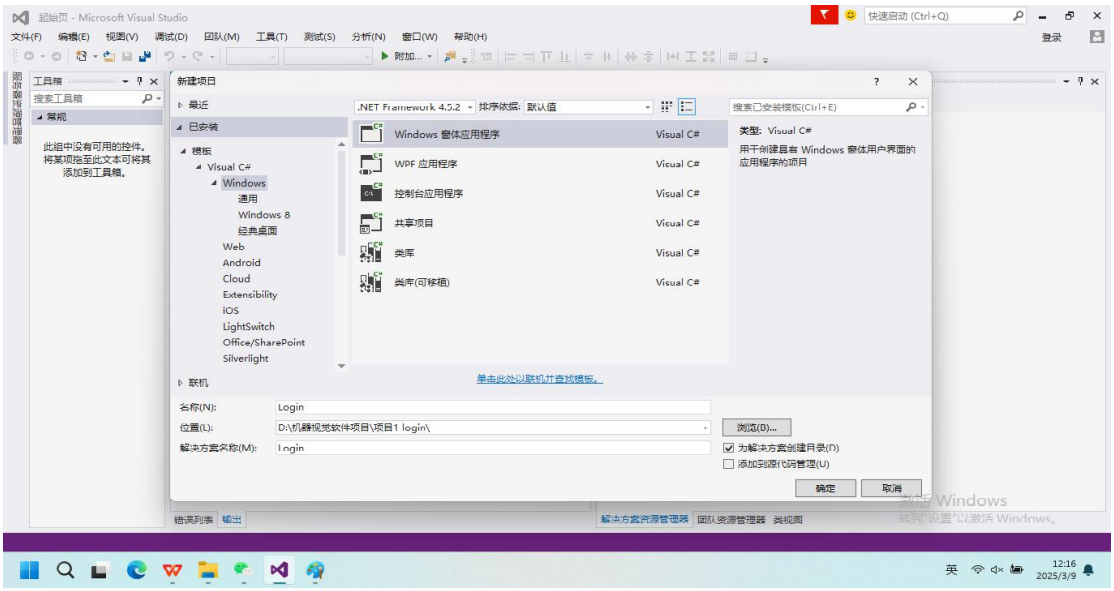


图 1-12 新建项目

设计界面。进入工具箱，讲相应的控件拖放在窗体上，设置各控件的属性。界面效果如图 1-15 所示，具体的控件属性，参考控件属性表 1-1。

表 1-1 登录窗体控件属性设置

控件名称	属 性	属 性 值
Label1	Name	iblUser
	Text	用户名
Label1	Name	iblPassword
	Text	密码
textBox1	Name	txtUser
textBox2	Name	Password
	PasswordChar	*
Button1	Name	btnLogin
	Text	登录
Button1	Name	btnCancel
	Text	取消
Form1	Name	frmLogin
	Text	用户登录
	Size	300，250

为了使得窗体的命名规范，可以对窗体进行重命名。如图 1-15 和图 1-16 所示，将 form1 窗体重命名为“frmLogin”。

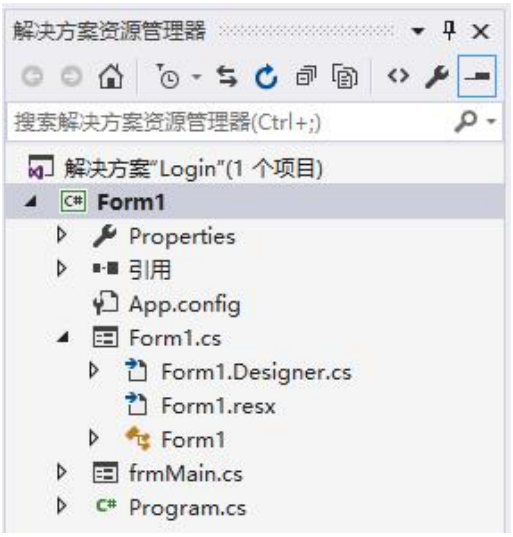


图 1-15 解决方案资源管理器

（4）添加第二个窗体。右击项目“Login”，在弹出的快捷菜单中选择“添加”，再选择“Windows 窗体”命令，如图所示，将出现添加新的 Windows 窗体的向导“添加新项”对话框，将第二个窗体命名为“frmMain”，如图 所示。

图 添加新的窗体

同样对 frmMain 窗体添加控件，具体的控件属性设置见表 1-2。

控件名称	属 性	属 性 值
Label1	Name	lblWelcome
	Text	欢迎进入机器视觉软件登录页面
frmMain	Text	主窗体
	Size	414，300

1.2.3 编写代码：

在 frmLogin 窗体中，双击“登录”按钮，进入该按钮的单击事件，编写代码。

```
private void btnLogin_Click(object sender, EventArgs e)
        {
            if (txtUser.Text == "admin" && txtPassword.Text == "user")
                frmMain frmMain1 = new frmMain();
        }
```

双击“取消”按钮，进入该按钮的单击事件，编写代码。

```
private void btnCancel_Click(object sender, EventArgs e)
        {
            txtUser.Text == " " ;
            txtPassword.Text=" " ;
            txtUser.Focus();
        }
```

(5) 调试程序。单击“生成”“生成解决方案”，或者没有错误，就可以单击“调试”在点击启动“启动调试”命令，或者按 F5，也可以启动调试。

项目 2 机器视觉打开相机程序的编写

项目知识目标

- 理解如何使用 Visual C# 打开并访问 USB 相机
- 掌握在 Visual C# 中使用第三方库进行图像捕捉
- 熟悉相机的设备管理和资源释放

项目能力目标

- 能够在 Visual C# 环境中使用编程代码访问并控制 USB 相机
- 能够使用第三方库捕捉视频流并展示到窗口
- 能够正确处理相机设备，确保资源的正确释放

任务 2.1 准备开发环境

学习通扫一扫 看视频

2.1.1 安装必要的第三方库 OpencvSharp3.5

打开 Visual Studio，创建一个新的 Windows 窗体应用程序项目。在菜单栏中选择“项目”，在选择下拉菜单中“管理 NuGet 程序包”，解决方案资源管理器中，右键点击项目并选择“管理 NuGet 程序包”，在搜索栏中输入 OpencvSharp3，系统回自动查找 OpencvSharp3 的库，找到之后选择 OpencvSharp3-anycpu 库安装。

2.1.2 设计界面

进入工具箱，讲相应的控件拖放在窗体上，设置各控件的属性。界面效果如图 2-3 所示，具体的控件属性，参考控件属性表 1-1。

控件名称	属 性	属 性 值
Button1	Name	btnOpen
	Text	打开摄像头
Button2	Text	btnClose
	Text	关闭摄像头
PictureBox1	Name	PictureBox
	Text	视频窗口

图 2-1 程序界面

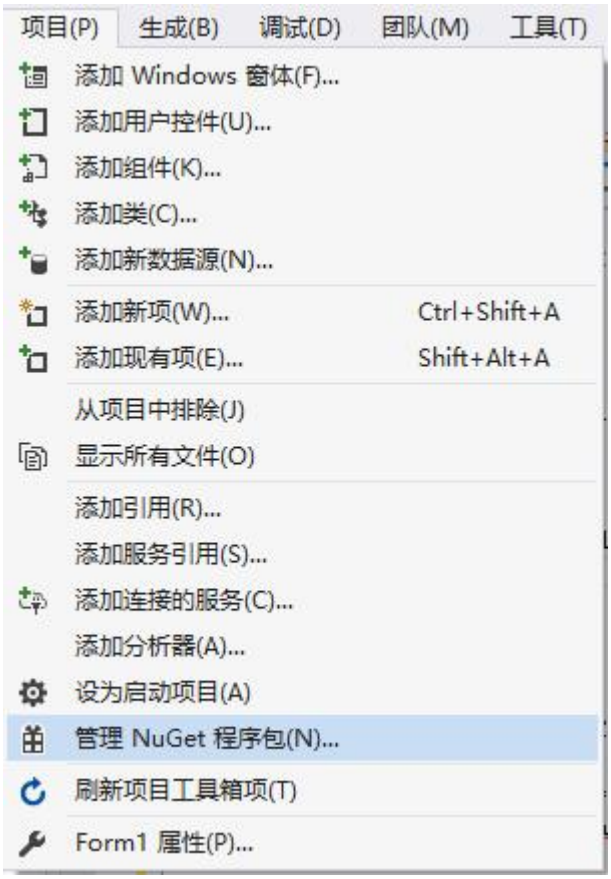


图 2-1 管理 NuGet 程序包

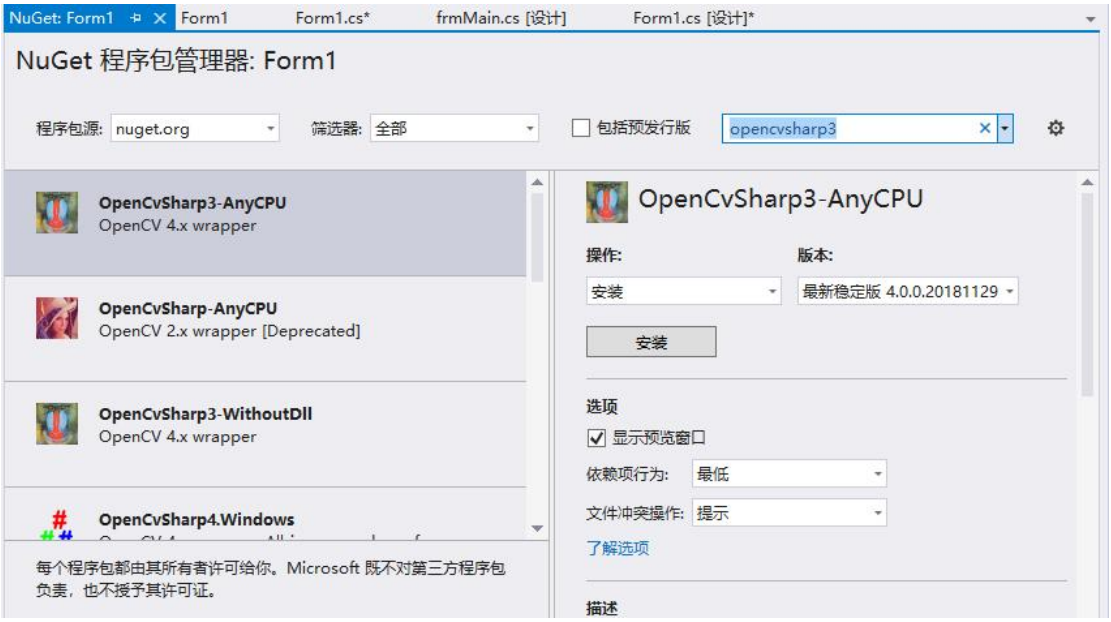


图 2-2 安装 OpencvSharp 库

图 2-3 程序界面

2.1.3 代码编写

1. 导入必要的命名空间

在代码文件顶部，导入以下命名空间：

```
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;
using OpenCvSharp;
using OpenCvSharp.Extensions;
using System;
using System.Windows.Forms;
```

2. 初始化并列出所有可用的 USB 相机设备

在窗体的“Load”事件中，初始化视频捕捉设备，并列出所有可用的 USB 相机：

```
 FilterInfoCollection videoDevices; // 存储视频设备
VideoCaptureDevice videoSource;    // 捕捉设备
    private void Form1_Load(object sender, EventArgs e)
{
    // 获取所有可用的USB视频设备
    videoDevices=new
      filterInfoCollection(FilterCategory.VideoInputDevice);
    // 确保存在至少一个视频设备
    if (videoDevices.Count == 0)
    {
        MessageBox.Show("没有检测到可用的视频设备！");
        return;
    }
    // 选择第一个设备并初始化捕捉设备
videoSource = new VideoCaptureDevice(videoDevices[0].MonikerString);
```

```
    // 设定捕捉数据流的事件处理方法
videoSource.NewFrame+= new NewFrameEventHandler(videoSource_NewFrame);
}
```

3. 处理相机捕捉的每一帧

创建一个方法来处理相机每一帧的视频数据，并将其显示到窗体上的 PictureBox 控件：

```
    private void videoSource_NewFrame(object sender, NewFrameEventArgs eventArgs)
{
    // 将捕捉到的每一帧图像显示到 PictureBox 控件中
    pictureBox1.Image = (Bitmap)eventArgs.Frame.Clone();
}
```

4. 启动和停止视频捕捉

在窗体的方法中启动视频捕捉设备，并在窗体关闭时停止捕捉：

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (videoSource.IsRunning)
    {
        // 停止视频捕捉
        videoSource.SignalToStop();
        videoSource.WaitForStop();
    }
```

```
}
```

第三步：资源管理

1. 释放资源

捕捉完毕后，确保正确释放视频设备资源，防止内存泄漏或设备占用问题。可以在窗体的 FormClosing 事件中处理：

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (videoSource.IsRunning)
    {
        videoSource.SignalToStop();
        videoSource.WaitForStop();
    }
    videoSource = null;
    videoDevices = null;
}
```

第四步：调试与运行

1.调试和测试

- 运行程序并确保 USB 相机正确连接到计算机。
- 按下启动按钮，确保图像在 PictureBox 中正常显示。
- 按下停止按钮，确保视频流停止显示。

2. 错误处理

- 在使用设备时，可能会出现设备连接问题或其他异常。可以在代码中添加异常处理，确保程序稳定运行：

```
 try
{
    videoSource.Start();
}
catch (Exception ex)
{
    MessageBox.Show("无法启动视频设备: " + ex.Message);
}
```

总结

通过上述步骤，我们在 Visual C# 中实现了 USB 相机的打开、视频流捕捉和显示。你学习了如何使用 OpencvSharp3-anycpu 库来访问 USB 相机设备、处理视频流数据，并在窗体中展示实时视频。这个过程展示了设备管理和图像捕捉的基本原理，帮助你掌握如何使用外部设备进行图像和视频处理。

项目 3 机器视觉图像形态处理

项目知识目标

- 学习如何在 C# 环境中使用 OpenCvSharp3 库来进行图像处理，包括如何导入、处理和保存图像。
- 了解图像缩放算法，如最近邻插值、双线性插值和立方插值等，学会在实际开发中选择合适的算法进行图像缩放。
- 学会通过 OpenCvSharp3 获取和修改图像的像素数据，以及如何使用图像处理技术（如缩放）来调整图像尺寸。

项目能力目标

- 1. 能够在 C# 项目中安装并配置 OpenCvSharp3 库，成功加载图像。
- 2. 能够使用 OpenCvSharp3 进行图像缩放，并能根据需求选择不同的插值算法。
- 3. 能够通过代码实现图像的保存，确保图像处理后不丢失数据，并输出成所需的格式。

任务 3.1 图像的缩放程序的编写

学习通扫一扫 看视频

3.1.1 实现的效果

设计一个机器视觉图像缩放界面，设置两个按钮，分别用于打开要缩放的图片和执行缩放操作。当用户点击打开图片按钮上，选择要打开的图片，点击执行缩放后，可以看到图像进行了缩放（50%）比例。执行效果如图 3-1。

3.1.2 图像缩放算法介绍

使用 OpenCvSharp3 提供的 Cv2.Resize 方法进行图像缩放。可以根据需要选择不同的缩放尺寸和插值算法。这里有几种常见的插值方法：

最近邻插值(InterpolationFlags.Nearest)

双线性插值 (InterpolationFlags.Linear)

立方插值 (InterpolationFlags.Cubic)

拉格朗日插值 (InterpolationFlags.Lanczos4)

3.1.3 安装必要的第三方库 OpencvSharp3.5

打开 Visual Studio，创建一个新的 Windows 窗体应用程序项目。在菜单栏中选择“项目”，在选择下拉菜单中“管理 NuGet 程序包”，解决方案资源管理器中，右键点击项目并选择“管理 NuGet 程序包”，在搜索栏中输入 OpencvSharp3，系统回自动查找 OpencvSharp3 的库，找到之后选择 OpencvSharp3-anycpu 库安装。

进入工具箱，讲相应的控件拖放在窗体上，设置各控件的属性。界面效果如图 2-3 所示，具体的控件属性，参考控件属性表 3-1。

控件名称	属 性	属 性 值
Button1	Name	btnOpen
	Text	打开摄像头
Button2	Text	btnClose
	Text	关闭摄像头
PictureBox1	Name	PictureBox
	Text	视频窗口

3.1.4 代码编写

```
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;
using OpenCvSharp;
using System.IO;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }


        Mat dstimg = new Mat();//定义目标图片格式
        Mat srcimg = new Mat();//源图片格式

        Bitmap bitmap1;   //定义源图片放置在 picturebox 中文件的格式
        Bitmap bitmap2;  //定义目标文件放置在 picturebox 中文件的格式

        private void button1_Click(object sender, EventArgs e)
        {   string imgname = "";//定义一个需要打开的文件名，保存打开的图像
            OpenFileDialog OpenFileDialog1 = new OpenFileDialog();
```

```
            if (OpenFileDialog1.ShowDialog() == DialogResult.OK)
            {
                imgname = OpenFileDialog1.FileName;
             }
            else
            {
                MessageBox.Show("打开图片错误");
            }

            Mat srcimg = new Mat(imgname, ImreadModes.AnyColor);
            Cv2.Resize(srcimg, dstimg, new OpenCvSharp.Size(), 0.5, 0.5);
//opencvsharp 的尺寸转换
            Bitmap bitmap1 = OpenCvSharp.Extensions.BitmapConverter.ToBitmap(srcimg);//
把 MAT 格式图片转化为 bitmap 格式，才能在 picturebox 中显示
            pictureBox1.Image = bitmap1;//把图片显示在 picturebox 中，表示打开了一副图像

        }

              private void button2_Click(object sender, EventArgs e)
        {
           Bitmap bitmap2 = OpenCvSharp.Extensions.BitmapConverter.ToBitmap(dstimg);//
把 MAT 格式图片转化为 bitmap 格式，才能在 picturebox 中显示
            pictureBox2.Image = bitmap2; //把已经缩放了的图像显示在 picturebox2 中

        }
     }

    }
```

任务 3.2 图像任意角度的旋转与缩放比例可设定程序编写

3.2.1 实现的效果

设计一个机器视觉图像缩放和旋转的界面，设置两个按钮，分别用于打开要缩放的图片和执行缩放和旋转操作。同时定义两个图像控件，用于存放打开和操作之后的图像。当用户在旋转角度和缩放比例的文本编辑框内输入要旋转的角度和缩放的比率之后，点击开始操作按钮上，选择要打开的图片，点击执行操作后，可以看到图像进行了缩放和旋转比例。执行效果如图 3-2。

进入工具箱，讲相应的控件拖放在窗体上，设置各控件的属性。界面效果如图 2-3 所示，具体的控件属性，参考控件属性表 3-1。

控件名称	属 性	属 性 值
Button1	Name	btnOpen
	Text	选择要打开的图片
Button2	Text	btnClose
	Text	执行操作
PictureBox1	Name	PictureBox1
	Text	原图
PictureBox2	Name	PictureBox2
	Text	旋转和缩放后的图像

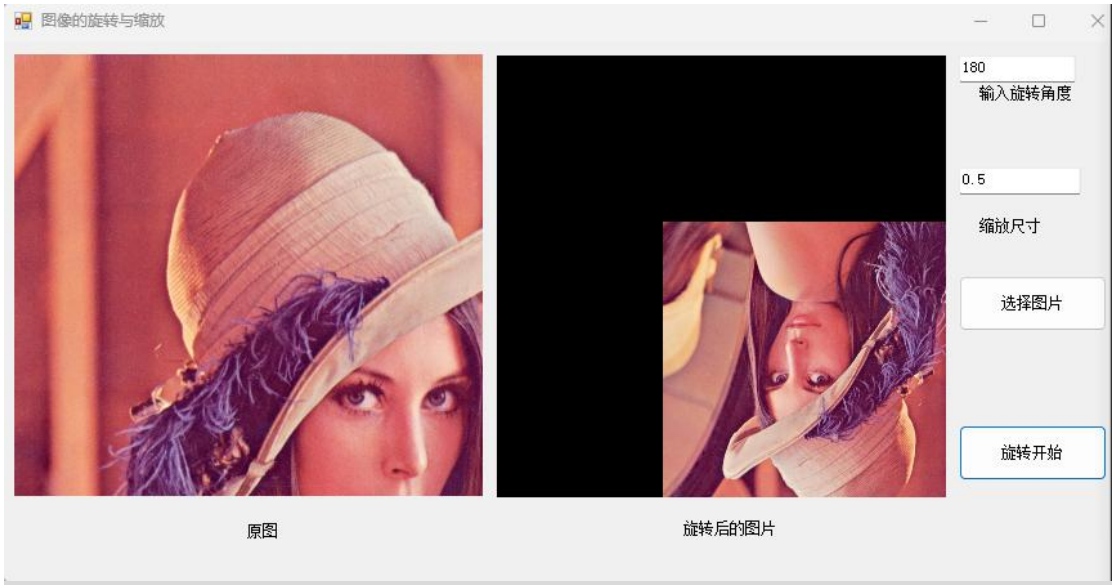


图 3-2 旋转和缩放界面

3.2.2 代码编写

```
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;
using OpenCvSharp;
using System.IO;
```

```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }



        Mat dstimg = new Mat();
        Mat srcimg = new Mat();
        Mat img45 = new Mat();

        Bitmap bitmap1;
        Bitmap bitmap2;
        Bitmap bitmap3;

        double weigh, height;



        private void button1_Click(object sender, EventArgs e)
        {

            string imgname = "";//定义一个文件名，保存打开的图像
            OpenFileDialog OpenFileDialog1 = new OpenFileDialog();
            if (OpenFileDialog1.ShowDialog() == DialogResult.OK)
            {
                imgname = OpenFileDialog1.FileName;
             }

            else
            {
                MessageBox.Show("打开图片错误");

            }

            Mat srcimg = new Mat(imgname, ImreadModes.AnyColor);

            Point2f Img_center = new Point2f(srcimg.Cols/ 2, srcimg.Rows/ 2);//设定旋转
中心点为原图像的中心点
```

```
            double angle = Convert.ToDouble(textBox1.Text);//获取文本框输入的数据作为旋转的角度值
            double scale = Convert.ToDouble(textBox2.Text);//获取文本框输入的数据作为缩放比率


            Mat R_mat = Cv2.GetRotationMatrix2D(Img_center, angle, scale);//安装输入的旋转中心，角度及缩放比率进行旋转

            Cv2.WarpAffine(srcimg,img45 , R_mat, srcimg.Size());

            Bitmap bitmap1 = OpenCvSharp.Extensions.BitmapConverter.ToBitmap(srcimg);//把 MAT 格式图片转化为 bitmap 格式，才能在 picturebox 中显示
            pictureBox1.Image = bitmap1;//把图片显示在 picturebox 中，表示打开了一副图像

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {

            Bitmap bitmap3 = OpenCvSharp.Extensions.BitmapConverter.ToBitmap(img45);//把 MAT 格式图片转化为 bitmap 格式，才能在 picturebox 中显示

            pictureBox3.Image = bitmap3;

        }


    }
}
```

通过这篇教程，学习了如何在 C# 中使用 OpenCvSharp3 库进行图像的缩放处理。了解了如何使用不同的插值算法进行图像缩放，并能够在实际开发中根据需求选择合适的算法。同时掌握了如何加载、缩放、保存和释放图像资源的基本操作。

项目 4 机器视觉图像边缘检测（Canny）

项目知识目标

- 学习 Canny 边缘检测算法的基本原理，包括如何通过梯度计算、非极大值抑制和双阈值方法提取图像的边缘信息。
- 熟悉如何在 Visual C# 环境中使用 OpenCvSharp3 库进行图像处理，特别是边缘检测的实现方法。
- 了解如何使用图像预处理（如灰度化、降噪等）提高 Canny 边缘检测效果，避免由于噪声和不均匀的亮度导致错误的边缘检测结果。

项目能力目标

- 1. 能够在 C# 项目中安装并配置 OpenCvSharp3 库，成功加载图像。
- 2. 能够应用 Canny 边缘检测算法，通过适当的参数调整检测效果。
- 3. 能够对 Canny 边缘检测的结果进行优化处理，提升边缘检测的精度和准确性。

任务 4.1 了解机器视觉 canny 边缘检测算法的原理

学习通扫一扫 看视频

Canny 边缘检测算法是由 John F. Canny 于 1986 年提出的，它通过一系列步骤来检测图像中的边缘，具有较好的抗噪声能力和边缘定位精度。Canny 边缘检测主要包括以下步骤：

1. 灰度化：将彩色图像转换为灰度图像，简化后续处理。

2. 高斯滤波：通过高斯滤波器平滑图像，去除噪声。

3. 梯度计算：使用 Sobel 算子计算图像的梯度，获取边缘的强度和方向。

4. 非极大值抑制：对梯度图像进行细化，去除非边缘部分。

5. 双阈值处理：通过两个阈值（高阈值和低阈值）检测边缘，最终确定边缘位置。

图 4-1　lena 原图

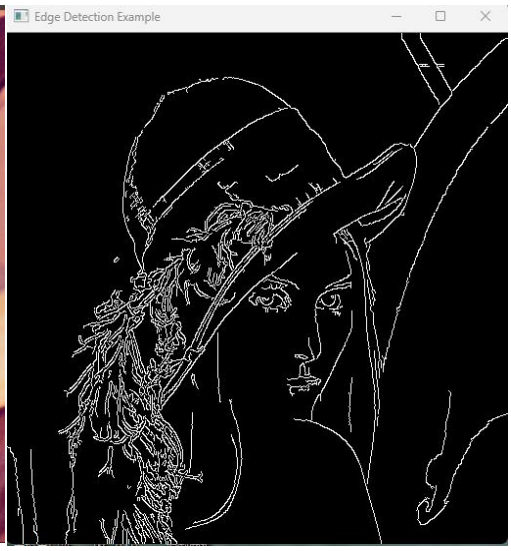


图 4-2　canny 边缘检测效果

任务 4.2 代码编写

```
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;
using OpenCvSharp;

namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
```

```
        private Mat image; // 存储原始图像
        private Mat edges; // 存储边缘检测结果图像
        public Form1()
        {

            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // 加载图像
            image = Cv2.ImRead("lena.jpg");

            // 创建窗口显示图像
            Cv2.NamedWindow("Edge Detection Example", WindowMode.AutoSize);
            Cv2.ImShow("Edge Detection Example", image);

            // 创建一个滚动条用于调整阈值
            trackBar1.Minimum = 0;
            trackBar1.Maximum = 255;
            trackBar1.Value = 100;
            trackBar1.TickFrequency = 10;
            trackBar1.Scroll += new EventHandler(trackBar1_Scroll);

            // 执行一次边缘检测
            DetectEdges();
        }

        private void DetectEdges()
        {
            // 使用滚动条当前值作为阈值进行边缘检测
            int threshold = trackBar1.Value;
            edges = new Mat(); // 创建新的 Mat 对象存储边缘检测结果
            Cv2.Canny(image, edges, threshold, threshold * 3); // 使用 Canny 函数进行边
缘检测
            Cv2.ImShow("Edge Detection Example", edges); // 在窗口中显示边缘检测结果
        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            DetectEdges();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
```

```
        {
            // 释放图像资源和窗口
            Cv2.DestroyAllWindows();
            image.Dispose();
            edges.Dispose();
        }
    }
}
```

通过这篇教程，学习了如何在 C# 中使用 OpenCvSharp3 库实现 Canny 边缘检测。掌握了如何进行图像预处理（如灰度化和高斯滤波），并通过调整参数（如阈值和滤波核大小）来优化边缘检测效果。此外，还学会了如何在图像处理后显示和保存结果，帮助实际开发中处理图像边缘检测任务。

项目 5 机器视觉图像点距离测量

项目知识目标

- 学习如何在图像中计算两个点之间的距离，理解欧几里得距离公式和其他距离计算方法。
- 学习如何在 OpenCvSharp3 中使用 `Point` 类和其他相关函数进行点的表示和计算
- 掌握图像的坐标系统，了解如何从图像中提取点坐标，并使用这些坐标进行距离测量。

项目能力目标

- 1. 能够通过 OpenCvSharp3 在图像中选择两个点，并计算它们之间的距离。
- 2.能够实现与图像相关的基本图形操作，如鼠标点击获取点坐标。
- 3. 能够在图像上标注点，计算并显示两个点之间的距离。

学习通扫一扫 看视频

5.1 实现的效果

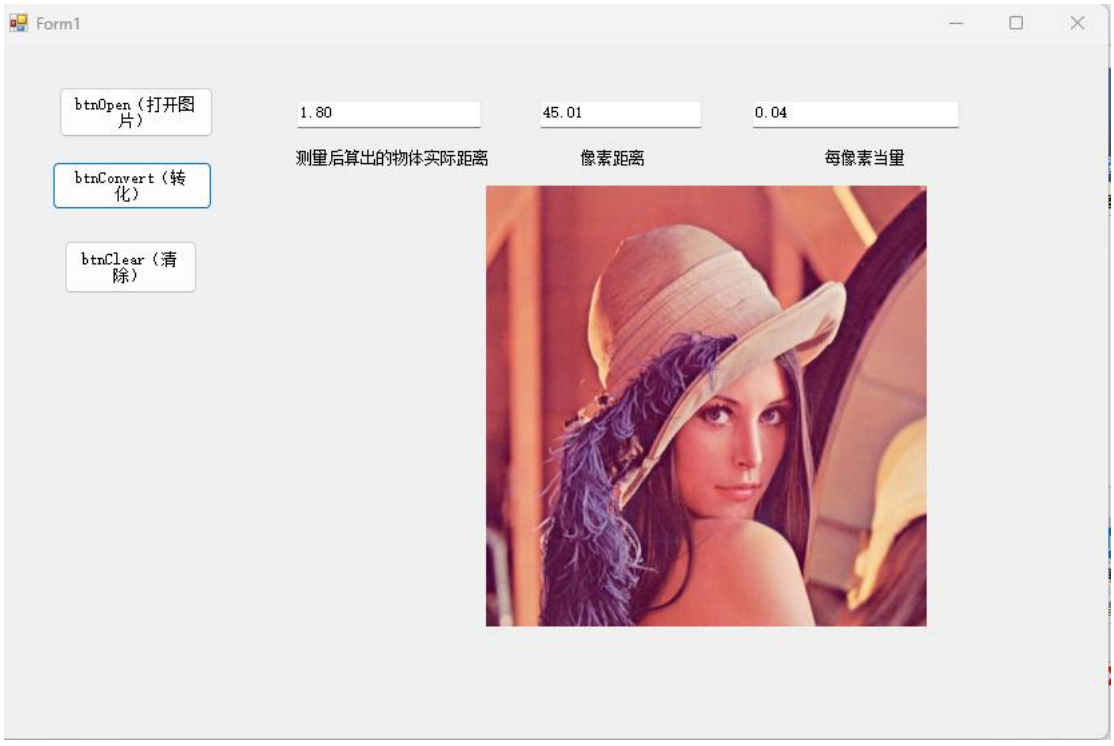
设计一个机器视觉图像点距离的界面，设置两个按钮，分别用于打开要缩放的图片和执行转换操作。同时定义 1 个图像控件，用于存放打开的图像。当用户在像素当量值之后，打开图片，用鼠标点击图像上的两个点，就可以测量出这两个点的距离。执行效果如图 5-1。

进入工具箱，将相应的控件拖放在窗体上，设置各控件的属性。界面效果如图 5-1 所示，具体的控件属性，参考控件属性表 5-1。

控件名称	属 性	属 性 值
Button1	Name	btnOpen
	Text	选择要打开的图片
Button2	Text	btnPlay
	Text	执行操作
PictureBox1	Name	PictureBox1
	Text	原图

5.2 图像点距离测量的原理

点距离计算原理 在计算机视觉中，图像通常由二维坐标系表示，坐标系的原点位于图像的左上角。每个像素点都有一个(x, y) 坐标，表示该像素的位置。欧几里得距离**是最常见的两点间距离计算方法。假设有两个点 (x1, y1)和(x2, y2)，它们之间的欧几里得距离 d 计算公式如下： d = sqrt{(x2 - x1)^2 + (y2 - y1)^2} 这个公式表示了两点之间的直线距离。



5.3 代码编写

```
using System;
using System.Drawing;
using System.Windows.Forms;
using OpenCvSharp;
```

```
namespace EdgeDetectionExample
{
    public partial class Form1 : Form
    {
        private Mat image; // 存储原始图像
        private Mat edges; // 存储边缘检测结果图像



        private void Form1_Load(object sender, EventArgs e)
        {
            // 加载图像
            image = Cv2.ImRead("your_image.jpg");

            // 创建窗口显示图像
            Cv2.NamedWindow("Edge Detection Example", WindowMode.AutoSize);
            Cv2.ImShow("Edge Detection Example", image);

            // 创建一个滚动条用于调整阈值
            trackBar1.Minimum = 0;
            trackBar1.Maximum = 255;
            trackBar1.Value = 100;
            trackBar1.TickFrequency = 10;
            trackBar1.Scroll += new EventHandler(trackBar1_Scroll);

            // 执行一次边缘检测
            DetectEdges();
        }

        private void DetectEdges()
        {
            // 使用滚动条当前值作为阈值进行边缘检测
            int threshold = trackBar1.Value;
            edges = new Mat(); // 创建新的 Mat 对象存储边缘检测结果
            Cv2.Canny(image, edges, threshold, threshold * 3); // 使用 Canny 函数进行边
缘检测
            Cv2.ImShow("Edge Detection Example", edges); // 在窗口中显示边缘检测结果
        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            // 滚动条数值变化时重新进行边缘检测
            DetectEdges();
```

```
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            // 释放图像资源和窗口
            Cv2.DestroyAllWindows();
            image.Dispose();
            edges.Dispose();
        }

    }
}
```

通过这篇教程，学习了如何在 C# 中使用 OpenCvSharp3 进行图像中的点距离测量。掌握了如何使用鼠标点击选择图像中的两个点，并通过计算欧几里得距离来得到它们之间的距离。通过这种方法，可以将点距离测量应用于实际的图像处理任务中，例如对象定位和运动分析等。

项目 6 机器视觉形态学运算

项目知识目标

- 学习如何在图像中计算两个点之间的距离，理解欧几里得距离公式和其他距离计算方法。
- 学习如何在 OpenCvSharp3 中使用 `Point` 类和其他相关函数进行点的表示和计算
- 掌握图像的坐标系统，了解如何从图像中提取点坐标，并使用这些坐标进行距离测量。

项目能力目标

- 1. 能够通过 OpenCvSharp3 在图像中选择两个点，并计算它们之间的距离。
- 2.能够实现与图像相关的基本图形操作，如鼠标点击获取点坐标。
- 3. 能够在图像上标注点，计算并显示两个点之间的距离。

任务 6.1 了解膨胀和腐蚀的原理

学习通扫一扫 看视频

图像处理中的形态学操作：膨胀和腐蚀，这两种技术主要用于改变二值图像中高亮区域的大小。膨胀使得物体边界向外扩展，而腐蚀则会消除边界点，减小物体面积。

1. 膨胀（Dilation）： 膨胀操作会“扩展”图像中的白色区域（通常是前景）。如果卷积核中有至少一个白色像素，目标像素位置就会变为白色。这样会导致前景区域变大。 常用于填补小的空洞或断裂，连接相近的物体，增强图像中的物体。

2. 腐蚀（Erosion）： 腐蚀操作会“收缩”图像中的白色区域。如果卷积核中没有完全覆盖白色区域，目标像素位置会变为黑色。这样会导致前景区域变小。常用于去除小的噪点，减小物体的边界，去除细小的连通部分。

1 膨胀和腐蚀介绍

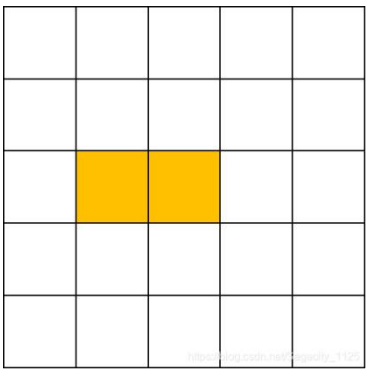
腐蚀和膨胀是形态学最基本的操作，都是针对白色部分（高亮部分）而言的。膨胀就是使图像中高亮部分扩张，得到比原图更大的高亮区域；腐蚀是原图像中的高亮区域被蚕食，得到比原图更小的高亮区域。膨胀是求结构元素下像素最大值，腐蚀是求结构元素下像素最小值。

膨胀（Dilation）和腐蚀（Erosion）是图像形态学中的两种基本操作，常用于处理二值图像（黑白图像）。它们的基本作用如下：

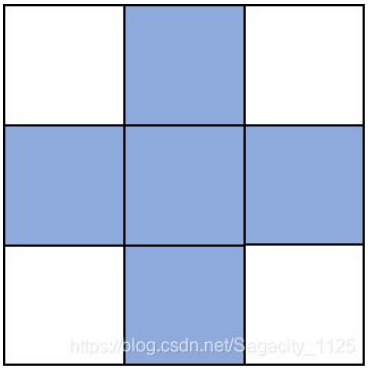
1.1 膨胀

具体操作：用一个结构元素扫描图像中的每一个像素，看结构元素覆盖下的原图的像素（二值图像只有 0 和 1）的最大值是多少，若最大值是 1，则该点像素为 1；若最大值是 0，则该点像素是 0。

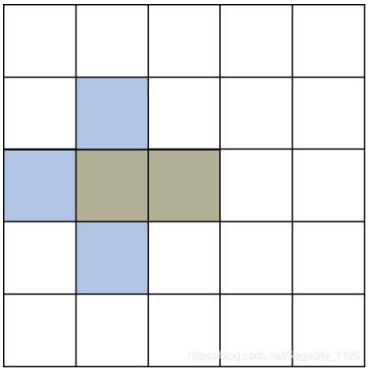
例子如下：
原图：



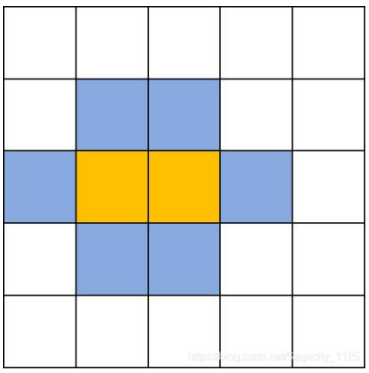
结构元素：



以第三行第二个像素点为例，将结构元素的中心点放置在该像素点上，结构元素覆盖的区域像素点的最大值为 1，因此第三行第二个像素点的值为 1。



膨胀后的结果：



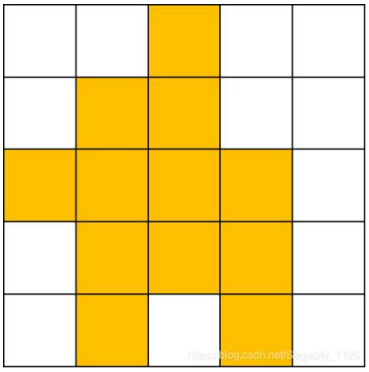
膨胀的作用是将与物体接触的所有背景点合并到物体中，使目标增大，可填补目标中的孔洞。

腐蚀

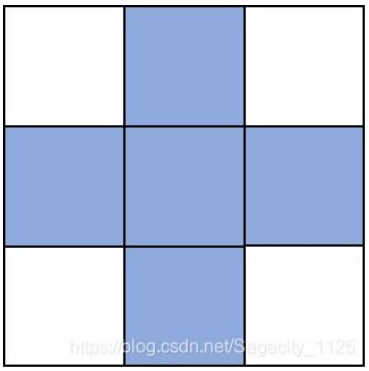
具体操作：用一个结构元素扫描图像中的每一个像素，用结构元素的中心对准当前正在扫描的像素，看结构元素覆盖下的原图对应的像素（二值图像只有 0 和 1）的最小值是多少，若最小值是 0，则该点像素为 0；若是 1，则该点像素是 1。

例子如下：

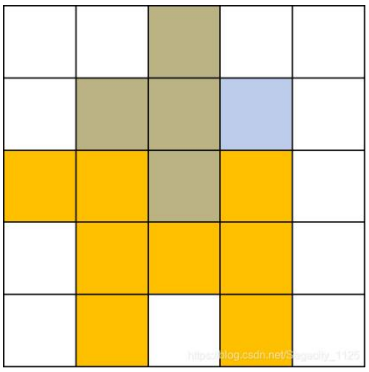
原图



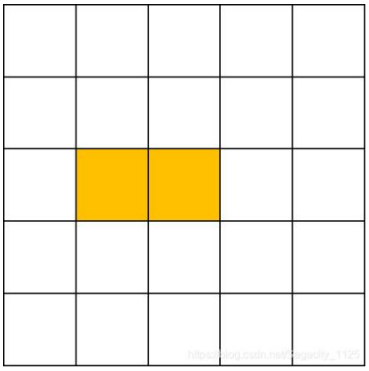
结构元素



以第二行第三个像素点为例，将结构元素的中心点放置在该像素点上，用结构元素的中心对准当前正在扫描的像素，结构元素覆盖的区域像素点的最小值为 0，因此第二行第三个像素点的值变成 0。



腐蚀后的结果：



腐蚀的作用是消除物体边界点，使目标缩小，可以消除小于结构元素的噪声点。

6.2 膨胀和腐蚀实现的效果

设计一个机器视觉膨胀和腐蚀操作的界面，设置 2 个按钮，用于打开腐蚀和膨胀的图片和执行操作操作。同时定义 1 个图像控件，用于存放打开的图像。当用户在下拉菜单中选择要膨胀和腐蚀操作后，点击执行操作，可以达到腐蚀和膨胀的效果。执行效果如图 6-1。

进入工具箱，将相应的控件拖放在窗体上，设置各控件的属性。界面效果如图 6-1 所示，具体的控件属性，参考控件属性表 6-1。

控件名称	属 性	属 性 值
Button1	Name	btnOpen
	Text	选择要打开的图片
Button2	Text	btnPlay
	Text	执行操作
PictureBox1	Name	PictureBox1
	Text	原图

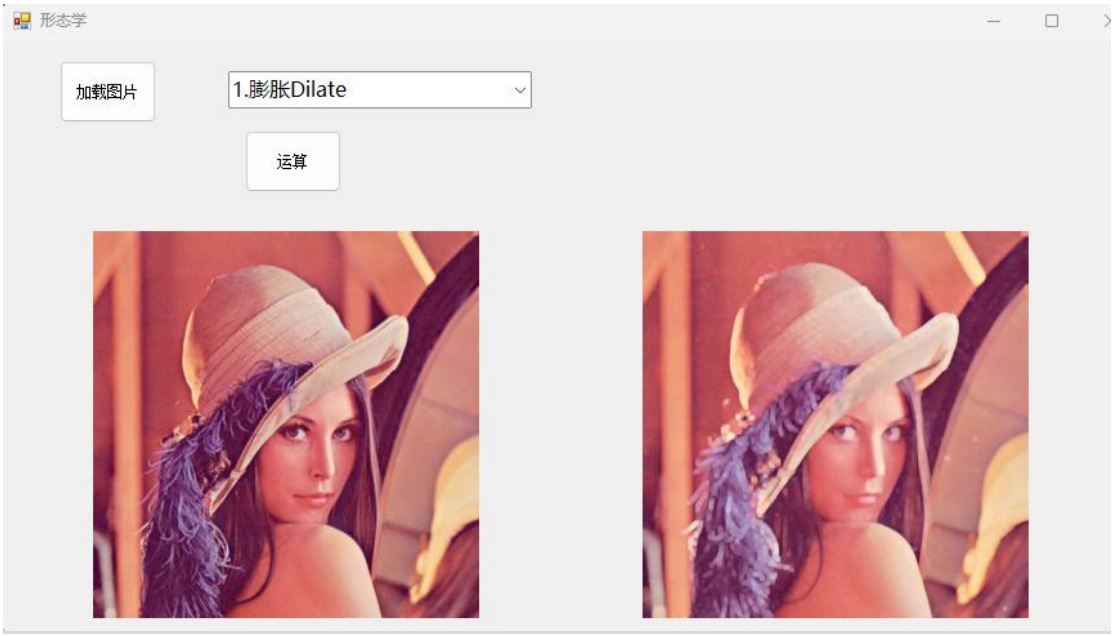


图 6-1 膨胀操作结果

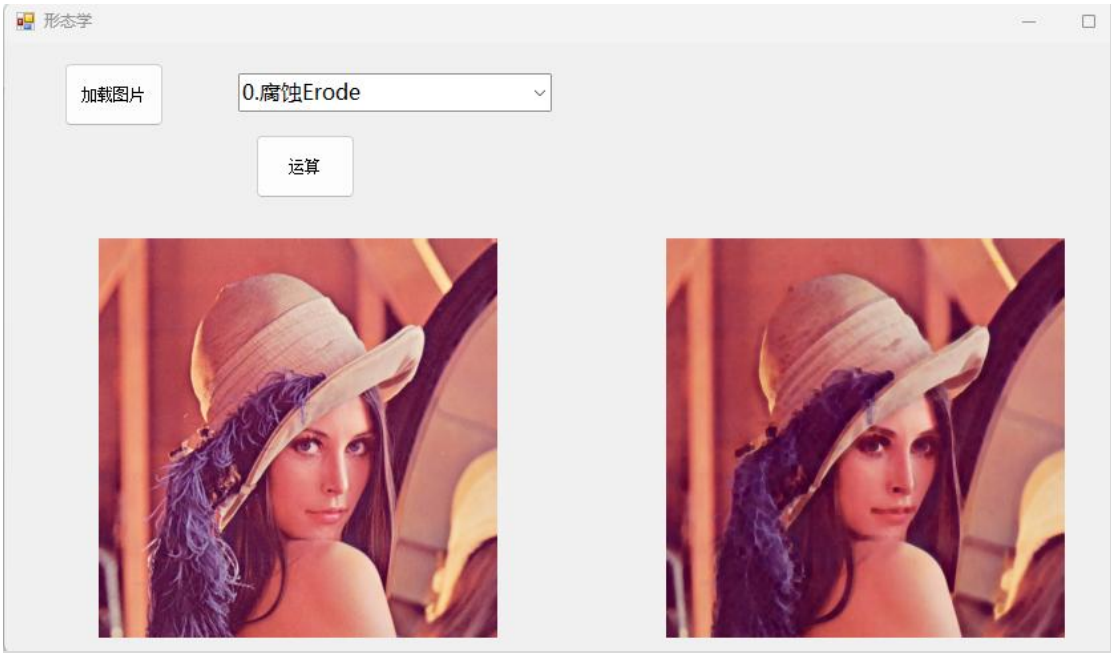


图 6-2 腐蚀操作结果

6.3 代码编写

```
using OpenCvSharp;
using System;
using System.Windows.Forms;
```

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

        private void Form1_Load(object sender, EventArgs e)
        {
            // 加载图像（灰度图）
            Mat image = Cv2.ImRead("image_path.jpg", ImreadModes.Grayscale);
            if (image.Empty())
            {
                MessageBox.Show("无法加载图像文件！");
                return;
            }

            // 创建一个 5x5 的矩形卷积核
            Mat kernel = Cv2.GetStructuringElement(MorphShapes.Rect, new Size(5, 5));

            // 执行膨胀操作
            Mat dilatedImage = new Mat();
            Cv2.Dilate(image, dilatedImage, kernel);
            Cv2.ImShow("Dilated Image", dilatedImage);

            // 执行腐蚀操作
            Mat erodedImage = new Mat();
            Cv2.Erode(image, erodedImage, kernel);
            Cv2.ImShow("Eroded Image", erodedImage);

            // 显示原图
            Cv2.ImShow("Original Image", image);

            // 等待按键
            Cv2.WaitKey(0);

            // 释放资源
            image.Dispose();
            dilatedImage.Dispose();
            erodedImage.Dispose();
        }
```

```
    }
}
```

通过这篇教程，了解膨胀和腐蚀操作的基本原理及其在图像处理中应用。学习了如何使用 OpenCvSharp3 库实现膨胀和腐蚀操作，并能够根据需要调整卷积核的大小和形状来优化图像处理效果。这些形态学操作广泛应用于图像预处理、去噪、分割等任务中。