毕业设计说明书

基于 OpenCV 的图像处理算法平台的开发与实现

学生姓名：何博轩　　学号：1907040314

学　　院：计算机科学与技术学院

专　　业：物联网工程　　班级：19070242

指导教师：柴锐

2023 年 6 月

基于 OpenCV 的图像处理算法平台的开发与实现

摘要

本文介绍了一个基于 Vue、Element Plus、Django 和 OpenCV 的图像处理算法平台的设计和实现。该平台实现了前后端分离，通过 API 接口和 JSON 格式的数据交换实现了前后端的交互。平台主要实现了图像上传、展示、处理和下载等功能，支持常用的图像处理算法，并通过应用进行了实现和测试。实验结果表明，我开发的图像处理应用具有良好的用户界面和稳定性，并且可以方便地进行扩展和优化。该图像处理应用的开发和实现对于促进图像处理技术的发展和应用具有重要意义。

本文首先分析了当前图像处理软件存在的问题，包括复杂性、功能限制和缺乏用户定制化等方面。然后，基于 OpenCV 这一广泛应用于图像处理的开源库，我设计和实现了一个图像处理平台。该平台具有直观的用户界面，使用户能够快速加载、编辑和处理图像，同时支持各种常见的图像处理操作。此外，平台还提供了丰富的定制化选项，允许用户根据自己的需求定义和应用自定义的图像处理算法。

本文的意义在于为用户提供了一个便捷、灵活且高效的图像处理平台。通过该平台，用户无需深入学习复杂的图像处理算法和编程技术，即可实现各种常见的图像处理任务。同时，平台的定制化功能也为有经验的用户提供了更大的灵活性和创造力空间，使其能够针对特定需求设计和应用定制的图像处理算法。这对于图像处理研究、教学和实际应用都具有重要意义，有助于提高图像处理技术的普及和应用水平。

关键词：opencv，vue，django，图像处理，平台搭建

Development and implementation of an OpenCV-based image processing algorithm platform

Abstract

This thesis presents the design and implementation of an image processing algorithm platform based on Vue, Element Plus, Django and OpenCV. The platform realises the separation of front and back end, and realises the interaction between front and back end through API interface and data exchange in JSON format. The platform mainly implements functions such as image upload, display, processing and download, supports common image processing algorithms, and has been implemented and tested through the application. The experimental results show that the image processing application I developed has a good user interface and stability, and can be easily extended and optimised. The development and implementation of this image processing application is of great significance in promoting the development and application of image processing technology.

This thesis first analyses the problems of current image processing software, including complexity, functional limitations and lack of user customisation. Then, based on OpenCV, a widely used open source library for image processing, I design and implement an image processing platform. The platform has an intuitive user interface that enables users to quickly load, edit and process images, while supporting a variety of common image processing operations such as filtering. In addition, the platform offers a wealth of customisation options, allowing users to define and apply custom image processing algorithms to suit their needs.

The significance of this thesis is to provide users with a convenient, flexible and efficient platform for image processing. The platform allows users to implement a wide range of common image processing tasks without the need for in-depth learning of complex image processing algorithms and programming techniques. At the same time, the platform's customisation capabilities provide greater flexibility and creativity for experienced users to design and apply customised image processing algorithms for specific needs. This has important implications for image processing research, teaching and practical applications, and will help to increase the popularity and application of image processing technology.

Key words: opencv, vue, django, image processing, platform building

目录

1 绪论

1.1 研究背景及意义

近年来，随着计算机视觉，模式识别，人工智能等技术的迅速发展，图像处理技术获得了前所未有的进步。本项目的研究成果可用于智能交通，工业检测，人脸识别，现代医疗等多个领域。

在信息时代快速发展的背景下，计算机技术在行业内的应用领域不断延伸，为各行业人员提供帮助，降低工作难度，提高效率。与此同时，在互联网时代，网民基数倍增，各种各样的海量数据信息也随之而来。当然，这些信息不都是有用的，里面也不乏一些冗余无用的信息，计算机处理这些冗余信息不免会降低运行效率，浪费可用资源。对此，图像处理作为计算机技术非常重要的学科分支之一，并作为一项比较先进的处理技术，对数据信息的优化具有不可轻视的优势地位。此技术可对图像信息进行识别处理，在处理质量和处理效率方面具有很大优势，并且处理过程方便快捷，同时可满足人们对图像的特殊处理需求。由于技术的优势突出，图像处理在各领域应用广泛，例如医学领域、交通领域、司法领域、军事领域、农业领域等[1]。

图像分割技术可以将图像分成若干个区域，实现对图像的目标提取和边界识别。图像描述方法可以通过提取图像的特征和属性，对图像进行表达和描述，例如形状、纹理、颜色等。图像分类技术可以将图像根据其内容进行分类和识别，常用于物体识别、人脸识别等领域。图像增强和复原工作可以通过滤波、去噪、增强对比度等方法，改善图像的质量和清晰度。伴随信息技术的革新和基础设施的完善，计算机图像处理技术也在不断更新，发展潜力巨大[2]。

1.2 国内外研究现状

随着数字图像技术的日益成熟，图像处理在当前研究中的重要性不断增加。作为人们感知外界重要信息的媒介[3]，图像已广泛应用于多个领域，包括灾难监视、医疗机器人的手眼系统、工业产品检测和检查，以及军事领域的精确制导等。

在过去几十年里，计算机图像识别技术取得了巨大进展。早期的研究主要集中在基于统计分类方法的二维图像识别技术上。然而，到上世纪 70 年代中期，图像处理技术取得了突破性进展，并得到了广泛应用。图像处理的目标是通过调整亮度、优化色彩以及进行旋转、平移、缩放等操作，从视觉上提升图像质量。此外，在计算机图像分析过

程中还需要提取一些重要的信息。为了方便存储和传输图像，还需要对图像进行数据变换、编码和压缩。

图像分割是一种重要的图像处理技术，通过将图像切分并提取关键信息，可以实现对图像局部特性的提取和分析，为后续的图像识别提供良好基础。图像分割技术在各个领域的应用和影响力不可忽视。

此外，为了减少图像数据的冗余信息，便于获取主要信息，我们采用图像编码和压缩技术对图像进行处理。这种方法旨在加快图像的传输和处理速度，同时减少存储开销，图像编码和压缩技术能够保证图像信息的完整性，确保原始信息不会丢失[4]。

由于人们对图像处理技术的要求越来越高，在计算机视觉与图像处理领域中已经有许多软件应用于图像处理。OpenCV 具有广泛的应用潜力，可在计算机视觉、形态学图像等领域发挥作用。

2 需求分析

2.1 项目概要

本系统将常用的图像处理算法整合到一个平台上，用户可以根据需要自定义选择操作。该系统适用于 RGB 三通道图像，并包含以下算法实现：阈值处理、边缘检测、平滑处理、图像分割、目标检测、视频多目标检测和车牌识别。此外，还提供图像均衡化、轮廓检测、直线检测和角点检测等功能。在平滑处理方面，系统提供多种滤波方法的实现。至于图像阈值处理，系统支持使用简单阈值进行处理。

2.2 功能需求

根据其实现的算法，具体有阈值处理、边缘检测、平滑处理、图像特征检测与匹配、图像分割、图片目标检测、视频多目标检测、车牌识别这些功能。

系统功能需求概要表如表 2.1 所示：

表 2.1 系统功能需求概要表

需求编号	需求内容
X1	搭建可视化界面
X2	前后端分离
X3	前后端数据通信
X4	打开本地图片，上传图片至系统
X5	阈值处理，将彩色图片转换为灰度图
X6	图像平滑
X7	车牌识别
X8	边缘检测
X9	直线检测
X10	图像分割
X11	角点检测
X12	图片目标检测
X13	视频流多目标检测
X14	图像均衡化
X15	轮廓检测

需求分析 X1 详细描述如表 2.2 所示：

表 2.2 需求分析 X1 详细描述

需求编号	需求内容
X1	用户选择处理方法。 调用本机系统文件，用户选择图片。 选择图像参数。 将图片上传至服务器。 服务器进行图像处理。 将处理好的图片返回前端。 前端展示处理好的图片。

需求分析 X2 详细描述如表 2.3 所示：

表 2.3 需求分析 X2 详细描述

需求编号	需求内容
X2	前端展示 ui 界面， 使用户方便操作。 后端接收数据，进行图像处理，并返回给前端处理好的图片 前端显示处理好的图片

需求分析 X3 详细描述如表 2.4 所示：

表 2.4 需求分析 X3 详细描述

需求编号	需求内容
X3	前端选择图像处理方法并调节参数。 后端接收数据。 后端进行图像处理后返回数据。 前端接收数据。 展示图片。

需求分析 X4 详细描述如表 2.5 所示：

表 2.5 需求分析 X4 详细描述

需求编号	需求内容
X4	可以调用系统文件选取功能，进行文件选取。

需求分析 X5 详细描述如表 2.6 所示：

表 2.6 需求分析 X5 详细描述

需求编号	需求内容
X5	选择阈值处理的类型，包括简单阈值、自适应阈值和 Otsu's 二值化 设置阈值分割值 点击处理按钮将数据传至后端。 后端根据数据进行图像处理并返回处理后图像。 前端接收图像并展示处理好的图像。

需求分析 X6 详细描述如表 2.7 所示：

表 2.7 需求分析 X6 详细描述

需求编号	需求内容
X6	选取要处理的图像。 选择处理参数。 后端根据数据进行图像处理并返回处理后图像。

需求分析 X7 详细描述如表 2.8 所示：

表 2.8 需求分析 X7 详细描述

需求编号	需求内容
X7	选取要处理的图像图像。 选择处理模型。 点击处理按钮将数据传至后端。 后端根据数据进行图像处理并返回处理后图像。 前端接收图像并展示处理好的图像。

需求分析 X8 详细描述如表 2.9 所示：

表 2.9 需求分析 X8 详细描述

需求编号	需求内容
X8	选取要处理的图像。 选择处理参数。 后端根据数据进行图像处理并返回处理后图像。

需求分析 X9 详细描述如表 2.10 所示：

表 2.10 需求分析 X9 详细描述

需求编号	需求内容
X9	选取要处理的图像。 选择处理参数。 后端根据数据进行图像处理并返回处理后图像。

需求分析 X10 详细描述如表 2.1 所示：

表 2.11 需求分析 X10 详细描述

需求编号	需求内容
X10	选取要处理的图像。 选择处理参数。 后端根据数据进行图像处理。 后端返回处理后图像。 前端显示图片。

需求分析 X11 详细描述如表 2.12 所示：

表 2.12 需求分析 X11 详细描述

需求编号	需求内容
X11	选取要处理的图像。 选择处理参数。 后端根据数据进行图像处理并返回处理后图像。

需求分析 X12 详细描述如表 2.13 所示：

表 2.13 需求分析 X12 详细描述

需求编号	需求内容
X12	选取要处理的图像。 选择处理参数。 后端根据数据进行图像处理 返回处理后图像。 前端显示图片

需求分析 X13 详细描述如表 2.14 所示：

表 2.14 需求分析 X13 详细描述

需求编号	需求内容
X13	选取要处理的图像。 选择处理参数。 后端根据数据进行图像处理并返回处理后图像。

需求分析 X14 详细描述如表 2.15 所示：

表 2.15 需求分析 X14 详细描述

需求编号	需求内容
X14	选取要处理的图像。 选择处理参数。 后端根据数据进行图像处理并返回处理后图像。

需求分析 X15 详细描述如表 2.16 所示：

表 2.16 需求分析 X15 详细描述

需求编号	需求内容
X15	后端根据数据进行图像处理并返回处理后图像。

2.3 系统功能结构

2.3.1 系统功能结构图

系统功能结构如图 2.1 所示：

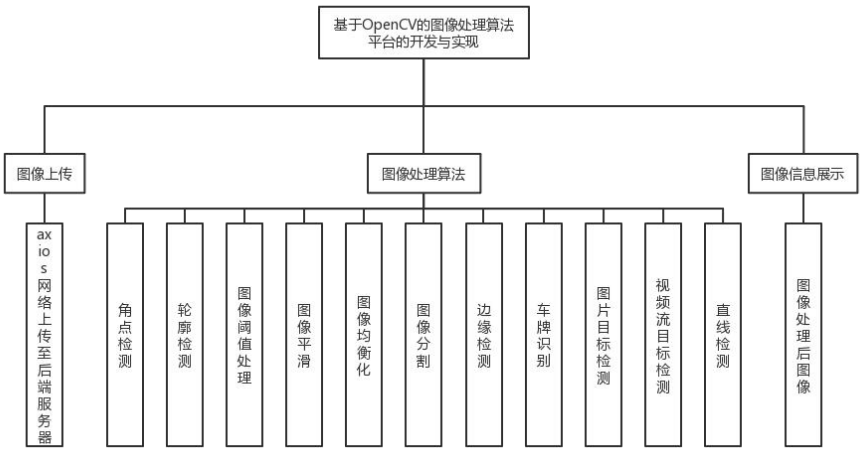

图 2.1 系统功能结构

系统将常用图像处理算法集合在一个平台上，使用 Web 可视化，使用户能够更加直观方便地进行相关操作。

本系统主要包含三部分, 图像的读入与保存, 图像信息展示以及处理图像常用算法。其中图像的读入保存模块，可以读取 RGB 三通道图。图像处理算法模块中集合的算法有以上十一种类型，其中包括阈值处理、边缘检测、平滑处理、图像分割、图片目标检测、视频多目标检测、车牌识别。图像均衡化、轮廓检测、直线检测、角点检测。

3 概要设计

3.1 开发环境

本系统所需环境配置如表 3.1 所示：

表 3.1 环境配置表

名称	版本	备注
macos 系统	13.3.1	系统环境
PyCharm	2023.1 arm	开发平台
Python	3.9.16	开发语言
OpenCV	4.7.0.72	算法处理
django	4.1.7	后端框架
vue	2.9.6	前端框架
element-plus	2.3.0	UI 框架
axios	1.3.5	网络请求
WebStorm	2023.1 arm	开发平台

3.2 算法描述

本系统实现的算法有：阈值处理、边缘检测、平滑处理、图像分割、图片目标检测、视频多目标检测、车牌识别。图像均衡化、轮廓检测、直线检测、角点检测。

具体描述如下：

3.2.1 图像平滑

平滑处理在图像处理中扮演去除噪声的角色，常被称为图像模糊[5]。然而，在进行平滑处理时，有时会面临处理效果不佳的情况，进而导致关键信息的模糊化。该处理方法的基本原理是使用周围相邻像素点的近似值替代噪声点的值[6]。

高斯滤波通过加权平均周围邻域像素的值，其中权重由高斯函数确定，以实现更加自然和柔和的平滑效果。这两种方法在不同场景下有不同的适用性和效果，用户可以根据需要选择合适的方法进行平滑处理。这两种算法都是常见的图像处理技术，能够有效地去除图像中的噪声，并改善图像的质量。

1）均值滤波

这种平滑处理方法可以用于图像的去噪和平滑化，减少图像中的噪声和不必要的细

节。通过平均邻域像素值的方式，可以使图像变得更加平滑和模糊，从而达到一定程度上的图像降噪效果。但需要注意平滑过程可能会导致图像细节的损失。图像为 f(x, y)，其中 x 和 y 分别表示像素的横纵坐标。数字图像中的每个像素点 g(x, y) 可以通过对领域内几个像素点的值进行平均计算得到。具体方法如式 3.1：

$$g(x,y)=\frac{1}{M}\sum_{(m,n)\in s} f(m,n) \quad \text{(式 3.1)}$$

其中，x, y=0, 1, 2, …, N-1;领域平均法在图像处理中的应用可以通过控制像素点领域的大小来有效控制图像的噪声。领域的大小可以被将（x, y）点周边点的座标定义为除（x, y）点外的座标集，该座标集中的座标点数为 M。利用域中的像素点平均法，可以得到该像素点的值。

只有当领域内的像素值与中心像素的差异小于阈值时，才会参与平均计算。通过增加阈值参数，可以削减模糊效应，保留更多的图像细节。

通过调整领域大小和阈值参数，可以在图像处理过程中平衡降噪效果和图像模糊程度之间的权衡。较小的领域和较低的阈值可以更好地保留细节，但可能无法有效降低噪声；较大的领域和较高的阈值可以更好地降低噪声，但可能导致图像模糊。因此，选择合适的参数是非常重要的，以满足具体应用场景的需求。具体方法如式 3.2：

$$g(x,y)=\begin{cases}\frac{1}{M}\sum_{(m,n)\in s} f(m,n), |f(x,y)-\frac{1}{M}\sum_{(m,n)\in s} f(m,n)| \geq T \\ f(x,y), otherwise\end{cases} \quad \text{(式 3.2)}$$

其中, T 为规定的非负阈值[7].

通过框选的区域内的所有像素点进行平均值计算，可以得到该点的像素值。这个计算过程可以视为将该区域内的像素点贡献出来，用于生成当前像素点的新值。通过求取平均值，可以有效地降低噪声的影响，使得图像变得更加平滑和清晰。

滤波核的大小决定了平均计算的范围，较小的滤波核则更加保留图像的细节，但可能对噪声的降低效果不如较大的核有效。

在实际应用中，我们需要根据具体情况选择适当的滤波核大小。这取决于图像的特性、噪声的程度以及对图像细节保留和噪声降低的需求。通过不断调整滤波核的大小和观察滤波结果，可以找到最佳的平衡点，以获得满足要求的图像处理效果。

通过框选的区域内的所有像素点进行平均值计算，可以得到该点的像素值。这个计算过程可以视为将该区域内的像素点贡献出来，用于生成当前像素点的新值。通过求取平均值，可以有效地降低噪声的影响，使得图像变得更加平滑和清晰。

在实际应用中，我们需要根据具体情况选择适当的滤波核大小。这取决于图像的特性、噪声的程度以及对图像细节保留和噪声降低的需求。通过不断调整滤波核的大小和观察滤波结果，可以找到最佳的平衡点，以获得满足要求的图像处理效果。在图像的特定位置，例如像素点为 226，我们选择了一个 5x5 的区域。通过对所选像素点进行求平均值的操作，可以得到像素值的变化情况，如图 3.1 所示：

23	158	140	115	131	87	134
238	0	67	16	247	14	220
199	197	25	106	156	159	173
94	149	40	107	5	71	171
210	163	198	226	223	156	159
107	222	37	68	223	157	110
255	42	72	250	193	75	184
77	150	17	248	197	147	150
218	235	106	128	65	197	202

→

23	158	140	115	131	87	134
238	0	67	16	247	14	220
199	197	25	106	156	159	173
94	149	40	107	5	71	171
210	163	198	133	223	156	159
107	222	37	68	223	157	110
255	42	72	250	193	75	184
77	150	17	248	197	147	150
218	235	106	128	65	197	202

图 3.1 均值滤波处理像素点变化

其中均值滤波的核如图 3.2 所示：

$$K = 1/25 \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

图 3.2 均值滤波核

2）高斯滤波

在前文提到的均值滤波中，计算每个像素值的平均值时，采用了相同权重的邻域像素值。然而，在高斯滤波中，像素中心点的权重较高，而离像素中心点较远的点的权重较低。这种原理使得邻域内每个像素值的权重不同，并且可以通过求取所有权重的和来实现。

3.2.2 边缘检测

边缘检测是一种用于发现图像中明显亮度变化的区域，并揭示物体的轮廓[8]的图像处理方法。

在图像中，边缘代表了像素灰度值或颜色发生明显变化的地方，通常对应着物体之间的界限或物体内部的结构变化。边缘检测的目标是准确地标记出这些边缘，并将其从图像的背景中分离出来。

通过观察图 3.3，我们可以清楚地看到具有灰度值为 4 和灰度值为 148 的像素之间存在明显的边缘。

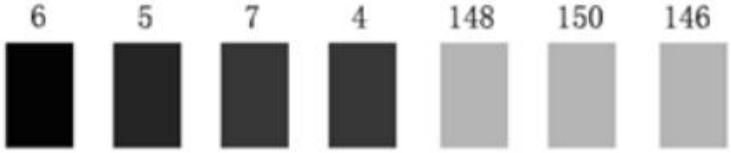

图 3.3 灰度值变化图

3.2.3 阈值处理

图像阈值处理是一种常用的图像处理技术，用于将图像的像素点分成不同的类别。它通过设定一个阈值，将图像中的像素点根据其灰度值与阈值的大小关系进行分类，实现对图像的分割[9]。下方的图 3.4 展示了某个图像的像素灰度分布，用于说明该方法的应用。

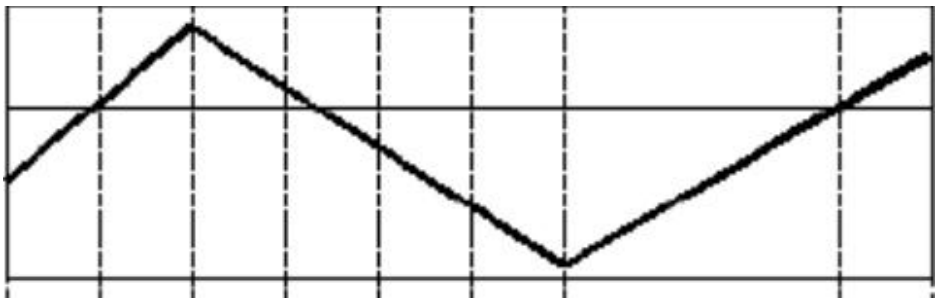

图 3.4 图像的像素灰度

在阈值处理中，简单阈值是常用的一种方法[10]。

1) 简单阈值

- 阈值二值化

$$dst(x,y)=\begin{cases} maxVal, & if\ src(x,y)>thresh \\ 0, & otherwise \end{cases} \quad (式\ 3.3)$$

如图 3.5 所示为阈值二值化后的图像像素灰度值：

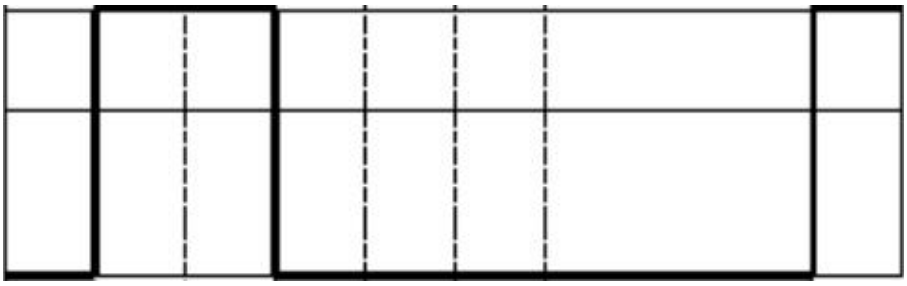

图 3.5 阈值二值化后的灰度值

- 阈值反二值化

$$dst(x,y)=\begin{cases}0, & if\ src(x,y)>thresh\\ maxVal, & otherwise\end{cases} \quad (式\ 3.4)$$

如图 3.6 所示为阈值二值化后的图像像素灰度值:

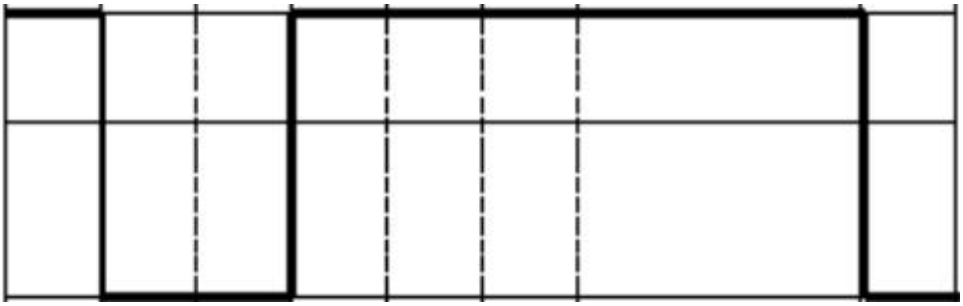

图 3.6 阈值反二值化后的灰度值

- 截断阈值化

$$dst(x,y)=\begin{cases}threshold, & if\ src(x,y)>thresh\\ src(x,y), & otherwise\end{cases} \quad (式\ 3.5)$$

如图 3.7 所示为阈值截断后的图像像素灰度值:

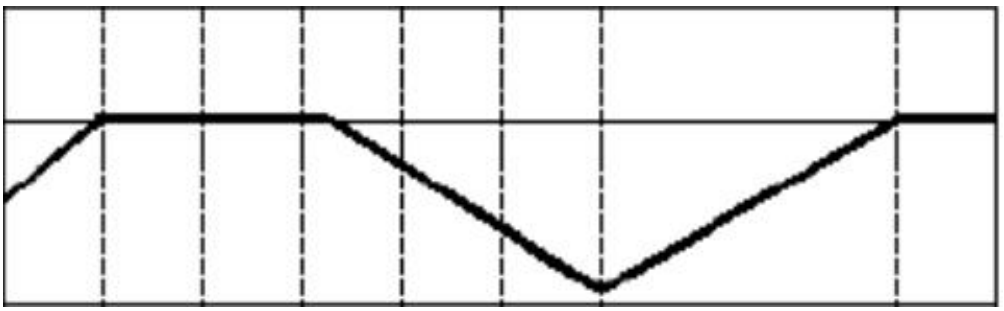

图 3.7 截断阈值后的灰度值

- 阈值取零

$$dst(x,y)=\begin{cases}src(x,y), & if\ src(x,y)>thresh\\ 0, & otherwise\end{cases} \quad (式\ 3.6)$$

如图 3.8 所示为阈值取零后的图像像素灰度值:

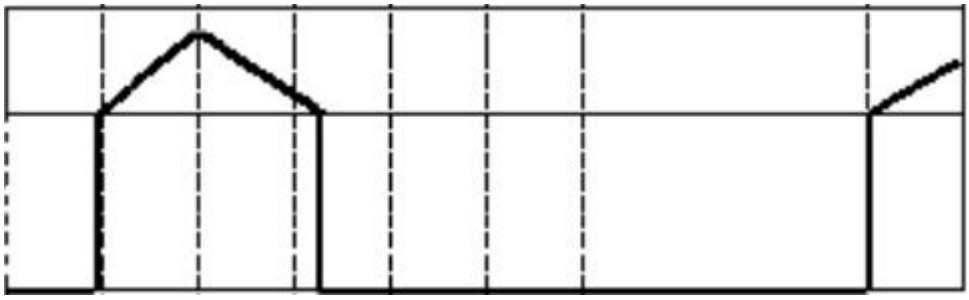

图 3.8 阈值取零后的灰度值

- 阈值反取零

$$dst(x,y)=\begin{cases}0, & if\ src(x,y)>thresh\\ src(x,y), & otherwise\end{cases} \quad (式\ 3.7)$$

如图 3.9 所示为阈值反取零后的像素灰度值:

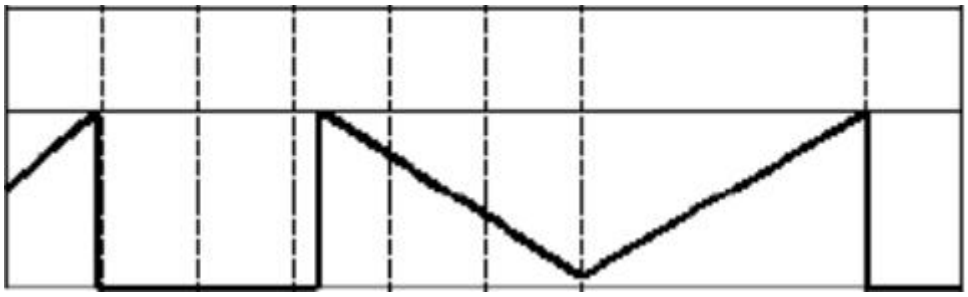

图 3.9 阈值反取零后的灰度值

3.2.4 图像分割

选择一个特定的图片作为需要加工的图片。并以此为依据，通过以下操作来具体实现：首先从前端请求中获得图像资料。执行图像模式的高效转换：将图像转成 HSV，从而达到去除噪声的目的，对于图像在系统中的应用和处理。它的代码是这样写的:

gray=cv2.cvtColor (img, cv2.COLOR_BGR2GRAY)

blurred=cv2.GaussianBlur (gray, (8, 8) , 0)

根据图像背景颜色，可以对图像进行二值化处理。二值化是将图像的像素值分成两种数值的处理方法，通常基于某个阈值将图像分为两个部分。

此外, 为了避免在图像处理过程中出现细节丢失的问题, 可以进行形态学细节处理, 形态学处理是基于图像腐蚀和膨胀原理进行的操作[10]。通过使用特定的卷积核对图像矩阵进行全面遍历，可以对图像进行细节处理。其中，图像腐蚀操作可以有效提取卷积核覆盖范围内的最小像素值。

通过结合二值化处理和形态学细节处理，可以对图像进行进一步的处理和分析，以满足具体应用的需求。被定义为:

$$ontpnt(x,y)=max\{input(x+x',y+y')\}(x',y')\in falseunit \quad (式\ 3.8)$$

图像膨胀可实现最大像素的有效提取[11] 被定义为:

$$output(x,y)=max\{input(x+x',y+y')\} \quad (式\ 3.9)$$

通常，在进行图像分割时，可以通过多种形式的形态学侵蚀和扩张来减少边缘对图像的干扰。为了精确地查找到图像的轮廓，并达到对图像的有效分割，进行了以下步骤的遍历替换：为一个大的（小的）；假想数字[i, j]=265；Ig [i, j]=0,0,0,265.在背景色中实现目标色的有效替代[12]。

3.2.5 图片目标检测

图片目标检测可以检测训练过的类名。图片检测可以分为静态图片和视频流目标检测，视频的图像检测相对更具挑战性，对算法和计算机性能提出了更高的要求[13]。本功能实现了基于静态图片的目标检测[14]。

3.2.6 车牌识别

图像预处理

首先，加载原始图片 1 并将其转换为灰度图像。这一步旨在减少数据量，将图像从 RGB 色彩空间转换为灰度色彩空间。

接下来，对灰度图像进行均值模糊处理。该处理可以柔化一些小的噪点，使图像更平滑。

然后，使用 Sobel 算子来提取图像中的垂直边缘。这是因为车牌通常具有许多垂直边缘，通过提取垂直边缘可以帮助我们更好地定位车牌区域[15]。

接着，将原始图片从 RGB 色彩空间转换为 HSV 色彩空间。这是因为车牌的背景色通常是蓝色或黄色，在 HSV 色彩空间中更容易提取出这些颜色。

在得到经过 Sobel 处理后的图像和 HSV 图像后[16]，利用图像中的蓝区和黄区，即可提取出牌照的区域。我们用索贝尔处理过的图像乘以 HSV 图像中的蓝、黄两个部分，以获得车牌区域的候选。

接下来，我们将候选区域进行二值化处理[17]，使用了最大类间方差的方法。此步骤将图片转化成黑白二进制图片，使车牌区域更加突出。

最后，对二值图像进行闭运算。闭合操作把牌照区域内的几条纵向边连在一起，进一步提取出完整的车牌区域。需要注意的是选择合适的核尺寸以获得较好的效果。

车牌定位中为了排除图像中的干扰项并尽可能保留车牌区域，我们可以进行以下步骤。

首先，获取图像中的轮廓。通过对二值图像进行轮廓检测，我们可以找到图像中所有的轮廓。

接下来，对每个轮廓求取其外接矩形。外接矩形是能够包围整个轮廓的最小矩形，可以通过计算矩形的长、宽以及长宽比等属性来进行筛选。

通过外接矩形的长，宽以及长宽比的等值关系，可以剔除掉一些非牌照的轮廓线，因为车牌的形状和尺寸具有一定的特征。

接下来，没有牌照的区域用背景颜色来进一步的排除。这里我们可以利用漫水填充算法，首先在外接矩形区域内生成种子点，种子点应为蓝、黄两色。之后，在经过填充的掩膜图像上画出外部矩形，并对外部矩形的大小进行判定，以确定其大小与牌照的要求相匹配。

漫水填充的目的有两个方面：一方面，预处理阶段另一方面，漫水填充。最终得到车牌区域。

最后，对获得的车牌区域进行仿射变换，校准其位置。这一步骤可以对车牌进行适当的旋转、缩放和平移，以得到更准确的车牌图像。

字符分割中为了提取车牌图像中的字符区域，我们可以进行水平投影和垂直投影两个步骤。

首先，进行水平投影。将二值化的车牌图像在 Y 轴方向上进行投影[18]，得到一条连续投影最长的段作为字符区域，由于车牌周围可能存在白色的边缘，我们可以过滤掉水平方向上的连续白线，以保留有效的字符区域，接下来，进行垂直投影，由于字符与字符之间通常会有一定的间隔，我们可以利用垂直投影作为水平分割的依据。通过对字符区域进行垂直投影，可以得到分割后的字符。为了保证分割的准确性，我们可以要求分割后的字符宽度达到平均宽度的阈值。通过水平投影和垂直投影的处理，我们可以将车牌图像中的字符区域提取出来。这些字符区域可以进一步用于字符识别或其他相关任务。

字符识别，在这一步中，我们的目标是识别出字符图像块，并输出车牌文本字符。

3.2.7 视频多目标检测

视频多目标检测与上述图片目标检测不同，此功能主要分为前端视频帧数获取，后端目检测别两部分。

前端目标识别，前端获取摄像头权限，将视频定时截取帧图像，发送到后端进行图像处理。

后端图像目标检测用了现在比较流行的 yolov8 官方模型，进行检测，检测后将目标检测相应数据返回前端。

前端接收数据使用 canvas 进行相应绘制，绘制的数据有，目标矩形方框，类名，置信度。

3.2.8 图像均衡化

将待处理图像记为 A，直方图分布如下图 3.4 纵坐标为$\mathrm{H_A(D)}$的直方图。

对待处理的图像 A 中的像素点执行单调非线性映射变换 $\mathrm{f: R \rightarrow R}$，变换后的图像记为 B，其直方图分布为下图中$\mathrm{H_B(D)}$所对应的图。

如下图 3.10 展示了整个过程：

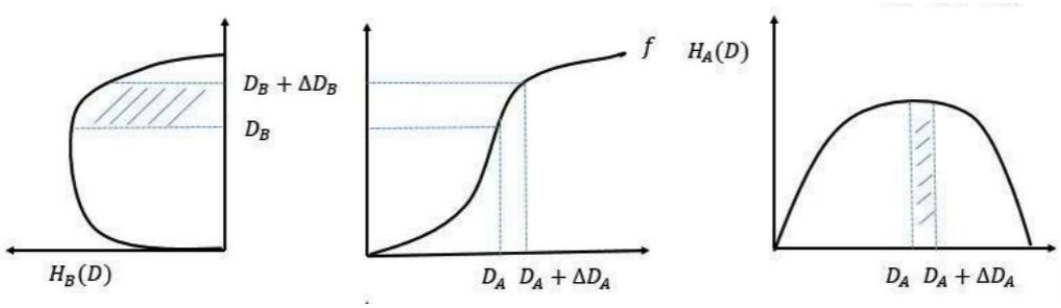

图 3.10 直方图均衡化原理

待处理图像 A 到 B 的单调非线性映射变换函数如上图 3.4 中右上方。

其中有：

$$D_B = f(D_A),\ D_B + \Delta D_B = f(D_A + \Delta D_A) \quad \text{(式 3.10)}$$

单调非线性变换函数f的作用是将处理前的 A 图像里面像素点灰度为D_A的全部变为D_B。

那么则有：

$$\int_{D_A}^{D_A+\Delta D_A} H_A(D)d(D) = \int_{D_B}^{D_B+\Delta D_B} H_B(D)d(D) \quad \text{(式 3.11)}$$

该公式特殊的有：

$$\int_0^{D_A} H_A(D)d(D) = \int_0^{D_B} H_B(D)d(D) \quad \text{(式 3.12)}$$

我们处理的目标是使得图像的直方图均匀分布，在理想情况下：

$$H_B(D) = \frac{A_0}{L} \quad \text{(式 3.13)}$$

其中，A_0为像素点个数，L为灰度级深度。于是，我们得到公式：

$$\int_0^{D_A} H_A(D)d(D) = \frac{A_0 D_B}{L} = \frac{A_0 f(D_A)}{L}, \text{则} f(D_A) = \frac{L}{A_0}\sum_{u=0}^{D_A} H_A(\mu) \quad \text{(式 3.14)}$$

4 详细设计

4.1 功能描述

4.1.1 阈值处理实现

1） 实现流程

该项目实现了基于 OpenCV 的阈值处理功能。以下是该功能的描述：

用户通过前端界面传递图像信息和阈值处理参数给后端，其中涉及阈值类型（threshold_type）、起始阈值（threshold_value）、最大阈值（max_value）等参数的选择。

具体实现过程如下：根据用户选择的阈值类型和定义的阈值参数，我们可以利用 OpenCV 中的 cv2.threshold()函数来进行阈值处理。该函数可以实现不同的阈值处理方式，如阈值二值化、阈值反二值化、截断阈值化、阈值取零、阈值反取零等。

通过调用 cv2.threshold()函数，并传入待处理的图像以及阈值参数，我们可以根据用户选择的阈值类型对图像进行阈值处理。函数将根据阈值类型和阈值参数将图像的像素值分成两个类别，并根据阈值处理方式对像素进行转换。

通过使用 cv2.threshold()函数进行阈值处理，我们可以根据用户选择的阈值类型和定义的阈值参数，对图像进行二值化、反二值化、截断阈值化等操作，从而满足不同的图像处理需求。具体的阈值类型对应的参数可以使用 cv2 模块中定义的常量。

阈值处理会将图像中的像素值与定义的阈值进行比较，根据阈值处理方法的规则将像素值转换为二值图像。处理后的结果以二值图像的形式返回。

最后，后端将处理后的二值图像使用 base64 编码方式返回给前端，前端进行解码并显示图像。

总结起来，该项目通过前后端的数据传递和解析，实现了基于 OpenCV 的阈值处理功能。用户可以选择不同的阈值处理方法和参数，通过后端的处理和图像处理算法，得到处理后的图像，并将其返回给前端进行显示。

4.1.2 边缘检测实现

该项目实现了基于 OpenCV 的边缘检测功能，其中仅采用了 Canny 边缘检测算法。以下是该功能的描述：

用户通过前端界面传递图像信息和边缘检测参数给后端，其中仅涉及边缘检测方法

参数的选择。

具体实现过程如下：后端解析前端传来的数据，包括图像信息和边缘检测参数。然后，将图像进行灰度转换，以便于边缘检测算法的处理。

接下来，使用 OpenCV 中的 cv2.Canny()函数进行 Canny 边缘检测。该函数需要设置三个阈值参数，分别为低阈值（low_threshold）、高阈值（high_threshold）、Sobel 算子的孔径大小（apertureSize）和是否使用精确的 L2 范数进行计算。这些参数可以由用户在前端界面定义。

Canny 边缘检测算法会根据这两个阈值对图像进行处理和是否使用精确的 L2 范数进行计算，并找出图像中的边缘。边缘的检测结果会以二值图像的形式返回。

最后，后端将处理后的二值图像使用 base64 编码方式返回给前端，前端进行解码并显示图像。

总结起来，该项目实现了基于 OpenCV 的边缘检测功能，仅采用了 Canny 边缘检测算法。用户可以通过前端界面传递图像和边缘检测参数，后端根据用户定义的参数进行 Canny 边缘检测，并将结果返回给前端显示。

4.1.3 平滑处理实现

该项目还实现了基于 OpenCV 的图像平滑处理功能，其中包括了均值滤波和高斯滤波。以下是该功能的描述：

用户通过前端界面传递图像信息和平滑处理参数给后端，其中涉及平滑处理方法的选择。

具体实现过程如下：后端解析前端传来的数据，包括图像信息和平滑处理参数。以便于平滑处理算法的处理。

接下来，根据用户选择的平滑处理方法，使用 OpenCV 中的相应函数进行图像平滑处理。如果用户选择了均值滤波，则可以使用 cv2.blur()函数，需要设置滤波器的大小（kernel_size）和边界处理方法（borderType）。

如果用户选择了高斯滤波，则可以使用 cv2.GaussianBlur()函数，需要设置滤波器的大小（kernel_size）、X 方向上的高斯核标准偏差（sigmaX），Y 方向上的高斯核标准偏差（sigmaY）和边界处理方法（borderType）。

均值滤波通过将像素周围的邻域像素进行平均来平滑图像，实现图像的模糊效果。高斯滤波则利用高斯函数的权重分配来平滑图像，较好地保留了图像的边缘信息。

其中边界处理方法包括：

cv2.BORDER_CONSTANT：使用常数填充边界。可以通过 borderValue 参数指定填充的常数值。默认情况下，常数值为 0。

cv2.BORDER_REPLICATE：复制边界像素的值。在这种模式下，如果滤波器超出图像边界，则使用最边缘的像素值进行填充。

cv2.BORDER_REFLECT：边界像素的值沿着边界镜像反射。如果滤波器超出图像边界，那么超出部分的像素值将通过反射方式进行填充。

cv2.BORDER_WRAP：边界像素的值按照环绕方式进行填充。超出图像边界的像素值将从相对位置的另一侧获得。

cv2.BORDER_TRANSPARENT：边界像素被视为透明，并且在滤波器计算过程中被忽略。

处理后的图像会根据选择的平滑处理方法以及设置的参数进行相应的平滑处理操作。

最后，后端将处理后的图像以 base64 编码的方式返回给前端，前端进行解码并显示图像。

总结起来，该项目通过前后端的数据传递和解析，实现了基于 OpenCV 的图像平滑处理功能。用户可以选择不同的平滑处理方法（如均值滤波和高斯滤波）和参数，通过后端的处理和图像处理算法，得到平滑处理后的图像，并将其返回给前端进行显示。

4.1.4 图像分割实现

该项目实现了基于 OpenCV 的图像分割功能。用户可以通过前端界面传递图像信息和分割参数给后端，其中仅涉及选择分割方法的参数。

具体实现过程如下：后端解析前端传来的数据，包括图像信息和分割参数。然后，将图像进行预处理。

接下来，使用 OpenCV 中的图像分割算法。分割方法的选择可以由用户在前端界面定义。

根据用户定义的分割参数和选择的分割方法，后端对图像进行分割处理，将图像中的不同区域或对象分离出来。分割的结果可以是一个二值图像、标记图像或具有不同颜色的分割结果图像，取决于所采用的分割算法和方法。

最后，后端将处理后的分割结果图像使用 base64 编码方式返回给前端，前端进行解

码并显示图像。

总结起来，该项目实现了基于 OpenCV 的图像分割功能。用户可以通过前端界面传递图像和分割参数，后端根据用户定义的参数和选择的分割方法进行图像分割，并将分割结果返回给前端显示。

4.1.5 目标检测实现

该项目实现了基于 OpenCV 的目标检测功能。用户可以通过前端界面传递图像信息给后端，后端利用预先训练好的模型进行目标检测处理。

具体的实现过程如下：后端首先解析前端传来的数据，包括图像信息。然后，利用已经训练好的模型对图像进行目标检测操作。

目标检测过程会对图像中的目标进行识别和定位，并将目标的位置和类别信息提取出来。后端可以根据具体的模型实现，使用相应的算法和技术来完成目标检测任务。

最后，后端将处理后的结果以二值图像的形式返回给前端，通常可以使用 base64 编码方式进行数据的传输和解码。前端接收到结果后可以进行解码，并将目标检测的结果显示在界面上，以便用户进行观察和分析。

4.1.6 车牌识别实现

该项目实现了基于 OpenCV 的车牌识别功能。用户可以通过前端界面将图像信息传递给后端进行处理。

具体的实现过程如下：后端首先解析前端传来的数据，包括图像信息。然后，利用训练好的模型对图像进行车牌识别操作。并进一步进行字符识别。

车牌识别过程中，后端会对图像进行预处理，例如调整图像尺寸、颜色转换等操作，以提高识别的准确性。然后，利用模型对预处理后的图像进行车牌区域的检测和字符识别，从中提取出车牌号码。

最后，后端将处理后的结果以二值图像的形式使用 base64 编码方式返回给前端，同时返回识别到的车牌信息。前端接收到结果后可以进行解码，并将图像和车牌信息显示在界面上供用户观察和使用。

4.1.7 视频多目标检测

该项目实现了基于 OpenCV 的目标检测功能。用户可以通过前端界面将视频信息传递给后端进行处理。

具体的实现过程如下：后端首先解析前端传来的数据，包括视频信息。然后，利用

训练好的模型进行目标检测操作。在本功能中，采用了 Yolov8 官方模型，该模型是一种流行的目标检测模型，能够快速且准确地识别图像中的目标。

目标检测过程中，后端会对视频进行逐帧处理，利用 Yolov8 模型对每一帧进行目标检测。模型会输出检测到的目标的坐标信息、类别名称和置信度等结果。后端将这些结果进行整理和提取，然后将它们返回给前端。

最后，前端接收到后端返回的结果后，可以根据坐标信息和类别名称等数据，在图像上绘制目标的边界框或其他可视化效果，以展示目标检测的结果。

4.1.8 图像均衡化

该项目实现了基于 OpenCV 的图像均衡化功能。用户可以通过前端界面传递图像信息给后端，进行图像均衡化处理。

后端解析前端传来的数据，包括包含图像信息的数据。

将图像进行色彩空间转换，使用 cv2.cvtColor()函数将图像从 RGB 色彩空间转换为灰度色彩空间。这是因为均衡化通常在灰度图像上进行。

使用 OpenCV 中的 cv2.equalizeHist()函数对灰度图像进行均衡化处理。该函数将直方图均衡化应用于图像，通过拉伸像素值的分布来增加图像的对比度。这可以帮助提高图像的细节和可视化效果。

均衡化后的图像将取代原始的灰度图像，成为处理后的结果。

后端将处理后的均衡化图像使用 base64 编码方式进行编码，然后返回给前端。

前端接收到编码后的图像数据，进行解码，并显示图像。

4.1.9 轮廓检测

该项目实现了基于 OpenCV 的轮廓检测功能。用户可以通过前端界面传递图像信息和边缘检测参数给后端，进行轮廓检测处理。

体实现过程如下：后端解析前端传来的数据，包括图像信息和边缘检测参数。然后，将图像进行灰度转换，以便于后续的阈值处理和轮廓检测算法的处理。

接下来，使用 OpenCV 中的 cv2.threshold()函数进行阈值处理。该函数需要设置一些参数，包度图像、阈值值（threshold_value）、最大像素值（max_value）和阈值处理类型（threshold_type）。该函数将根据设定的阈值将灰度图像二值化，得到二值图像。

然后，进行轮廓检测。该函数需要设置一些参数，包括轮廓检测模式（model）和轮廓近似方法（method）。该函数将在二值图像上寻找轮廓，并返回检测到的轮廓列表

（contours）以及轮廓的层次结构（hierarchy）。

阈值处理在上文有提到。

在轮廓近似方法中：

cv2.CHAIN_APPROX_NONE：将所有轮廓点都储存起来，并且使两个相邻的点之间的象素位差小于 1。该算法既保持了等高精度，又保持了等高精度。

cv2.CHAIN_APPROX_SIMPLE：压缩在水平，垂直和斜向上的元素，仅在那个方向上保持端点的坐标。比如，一条长方形的边框，仅需 4 个点就能保留边框的信息。通过简化轮廓形状，减少了点的数量，节省了存储空间。

最后，后端将轮廓检测的结果返回给前端。

总结起来，该项目实现了基于 OpenCV 的轮廓检测功能。用户可以通过前端界面传递图像和边缘检测参数，后端根据用户定义的参数进行灰度转换、阈值处理和轮廓检测，并将轮廓检测结果返回给前端。轮廓检测过程中使用的 model 和 method 参数的取值将根据具体需求确定。

4.1.10 直线检测

该项目实现了基于 OpenCV 的直线检测功能。用户可以通过前端界面传递图像和边缘检测参数给后端，进行直线检测处理。

后端解析前端传来的数据。将图像进行灰度转换，将图像转换为灰度图像。这是因为边缘检测通常在灰度图像上进行。

使用 OpenCV 中的 cv2.Canny()函数对灰度图像进行 Canny 边缘检测。该函数需要设置一些参数，如阈值参数。Canny 边缘检测算法会根据像素的梯度变化来检测图像中的边缘。

使用 OpenCV 中的 cv2.HoughLines()函数直线检测。函数将在边缘图像上应用霍夫变换，以检测可能的直线。返回的结果是检测到的直线的参数集合。

后端将直线检测的结果返回给前端，可以将参数集合作为数据传回给前端进行进一步的处理和显示。

总结起来，该项目实现了基于 OpenCV 的直线检测功能。用户可以通过前端界面传递图像和边缘检测参数，后端根据用户定义的参数进行 Canny 边缘检测和直线检测，并将直线检测结果返回给前端。

4.1.11 角点检测

该项目实现了基于 OpenCV 的角点检测功能。用户可以通过前端界面传递图像信息和边缘检测参数给后端，进行角点检测处理。

具体实现过程如下：后端解析前端传来的数据，包括图像信息和边缘检测参数。然后，将图像进行灰度转换，以便于角点检测算法的处理。

接下来，使用 OpenCV 中的 cv2.cornerHarris()函数进行 Harris 角点检测。该函数需要设置一些参数，包括邻域大小（blockSize）、Sobel 导数算子的孔径大小（ksize）和 Harris 角点检测参数（k）。该函数将在灰度图像上应用 Harris 角点检测算法，以检测图像中的角点。返回的结果是一个与输入图像大小相同的浮点型图像，其中角点位置被突出显示。

最后，后端将角点检测的结果返回给前端。

总结起来，该项目实现了基于 OpenCV 的角点检测功能。用户可以通过前端界面传递图像和边缘检测参数，后端根据用户定义的参数进行灰度转换和 Harris 角点检测，并将角点检测结果返回给前端。

4.1.12 后端设计

该项目旨在开发一个基于 OpenCV 的图像处理平台，使用 Django 作为后端框架，前端采用 Vue 和 Element Plus 进行开发。该平台旨在提供一系列图像处理功能，包括边缘检测、轮廓检测、目标检测等，以满足用户对图像处理的需求。

后端架构设计

后端采用 Django 框架进行开发，遵循 MVC（Model-View-Controller）设计模式，以实现清晰的架构设计和代码分离。

模型层（Model）：负责处理数据的存储和操作。在该项目中，模型层负责管理用户上传的图像数据、处理后的结果数据等。它可以定义数据模型、数据库表结构以及与数据库的交互操作。

视图层（View）：接收前端的请求，并根据请求调用相应的模型和算法进行图像处理。视图层负责处理业务逻辑，对图像进行图像处理，并将处理结果返回给前端。在 Django 框架中，视图层通常是通过定义函数或类来实现的。

控制器层（Controller）：负责处理用户请求的路由和逻辑控制。它接收前端的请求，根据请求的 URL 和 HTTP 方法将请求转发给相应的视图进行处理。控制器层处理用户

的输入，验证请求参数，处理异常情况，并将请求分发到适当的视图进行处理。

通过采用 MVC 设计模式，后端代码可以实现模块化和可维护性，模型、视图和控制器各自承担不同的责任，提高了代码的可读性和可扩展性。这种架构设计使得开发者能够更好地组织和管理代码，提高开发效率，并便于后续的维护和修改。

视图层（View）包含的后端接口如表 4.1：

表 4.1 后端接口

功能	接口函数
阈值处理	threshold(request)
边缘检测	canny(request)
平滑处理	smooth(request)
图像分割	image_segmentation(request)
目标检测	object_detection(request)
车牌识别	license_plate_recognition(request)
视频多目标检测	detect(request):
图像均衡化	calcHist(request)
轮廓检测	line(request)
直线检测	zline(request)
角点检测	points(request)

4.1.13 前端设计

前端架构设计，前端采用 Vue 和 Element Plus 进行开发，遵循组件化开发的原则，实现了可复用、可扩展的前端界面。

页面组件：包括图片上传组件，图片处理后组件，参数处理组件，通过路由进行切换和导航。

图片上传组件：提供图像上传功能。

图片处理后组件：提供处理结果展示功能。

参数处理组件：参数设置和开始处理功能，与后端进行数据交互。

其他辅助组件：包括侧边栏、提示框，提升用户体验。

1） 功能设计

图像上传功能：用户可以选择图像文件，并将图像上传至后端进行处理。

图像处理功能：平台提供多种图像处理功能。用户可以根据需求选择相应的处理功能，并设置参数。

数据交互设计中图像数据传递，前端发送数据，后端进行图像处理。

参数设置传递：用户在前端界面上设置图像处理的参数，通过 axios 请求将参数传递给后端，后端根据参数进行相应的图像处理操作。

5 测试

5.1 测试概要

基于 opencv 的图像处理平台的测试概要如表 5.1 所示:

表 5.1 测试概要

测试内容	说明
算法功能模块测试	针对算法的功能完整性的测试
上传图片测试	针对图片上传的测试
图片处理测试	针对图片处理参数以及数据传递的测试
图片下载测试	针对图片下载功能的测试
后端接收请求测试	针对后端能否正常接收请求的测试

5.1.1 算法功能测试

1）阈值处理

用户可以选择算法对应参数，上文有所叙述，测试结果如图 5.1：

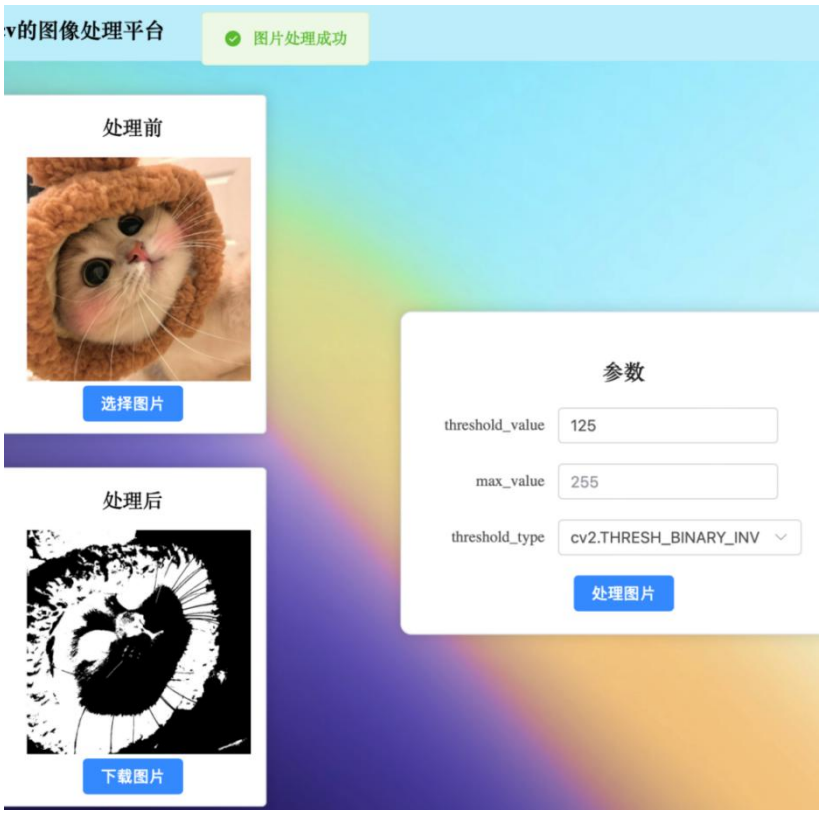

图 5.1 阈值处理模块测试

在这次测试中，我们选择起始阈值（threshold_value）参数为 125，最大阈值

(max_value) 参数为默认值 255, 阈值类型 (threshold_type) 为 THRESH_VINARY_INV。可以看到正确显示了阈值处理后的图像。

2）边缘检测

用户可以选择算法对应参数，上文有所叙述，测试结果如图 5.2：

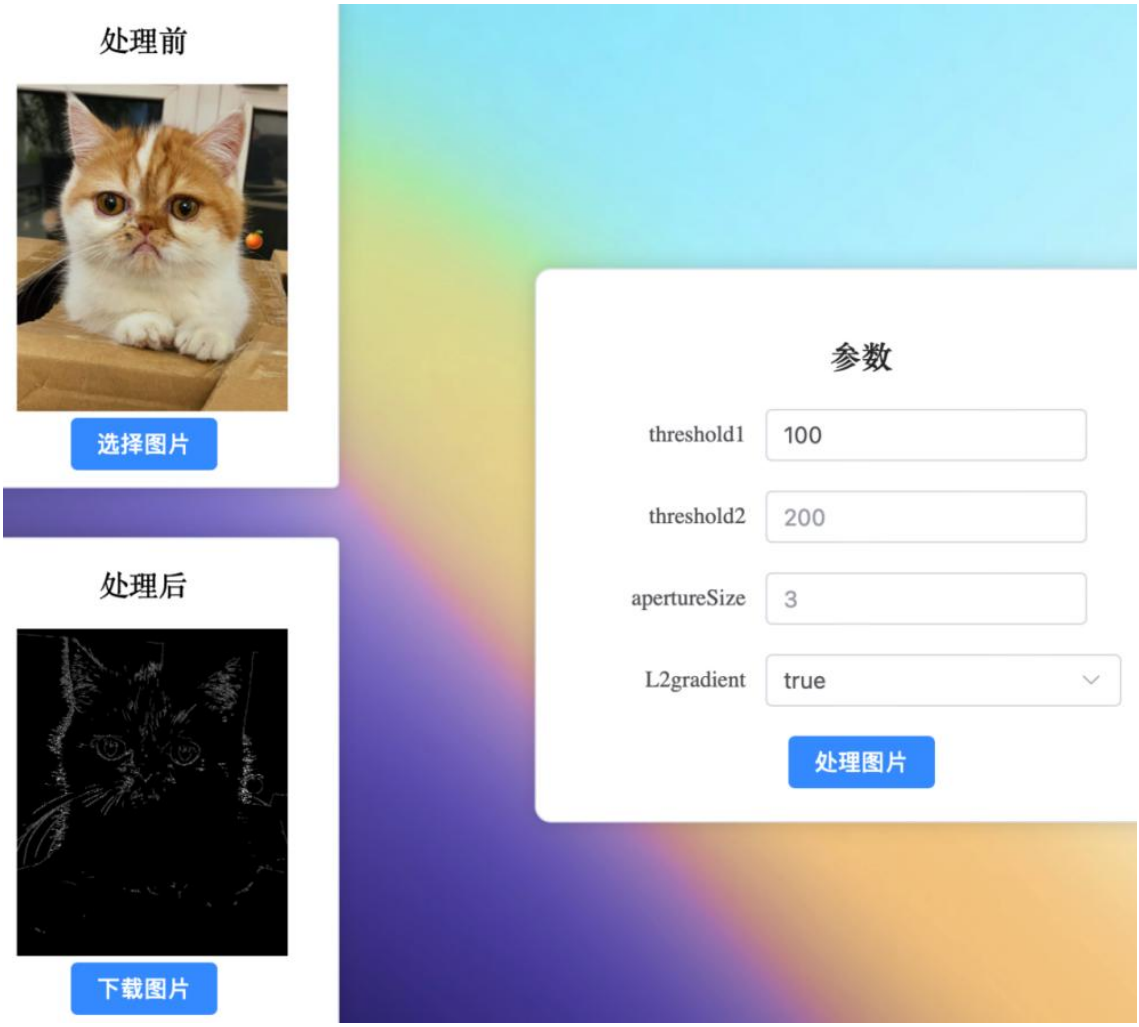

图 5.2 边缘检测模块测试

在这次测试中，我们选择的 threshold1 参数为 100、hreshold2 参数为默认 200，apertureSize 为默认的 3，并使用精确的 L2 范数进行计算（L2gradient），将参数设为 true。可以看到正确显示了边缘检测后的图像。

3）平滑处理

包括均值滤波和高斯滤波，用户可以选择算法对应参数，上文有所叙述，测试结果如图 5.3：

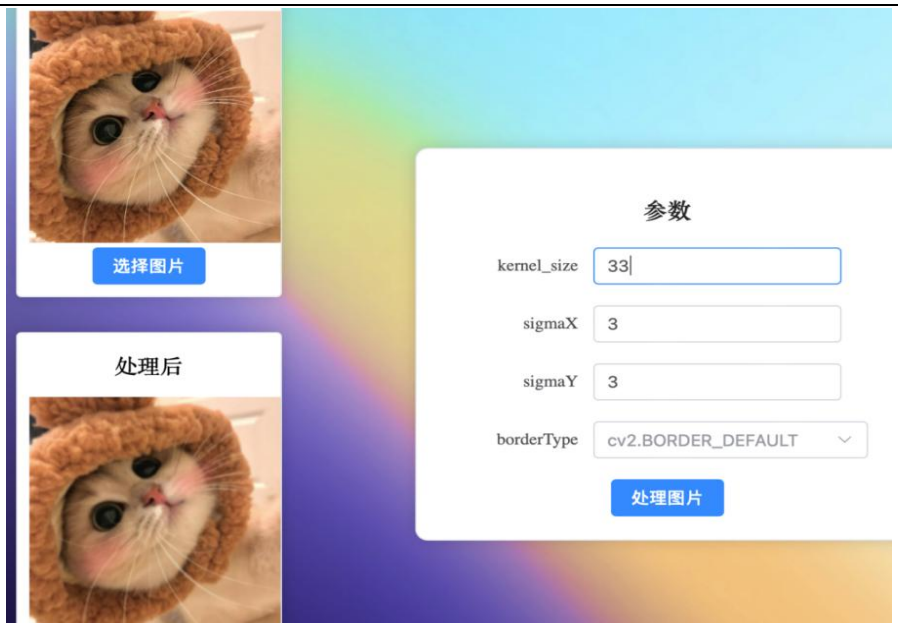

图 5.3 平滑处理模块测试

在这次测试中，我们选择的图像平滑方式为高斯滤波，选择的滤波器的大小（kernel_size）参数为 33、X 方向上的高斯核标准偏差（sigmaX）参数为 3，Y 方向上的高斯核标准偏差（sigmaY）参数为 3、边界处理方法（borderType）参数为默认的 cv2.BORDER_DEFAULT。可以看到正确显示了平滑处理后的图像。

4）图像分割

用户可以选择算法对应参数，上文有所叙述，测试结果如图 5.4：

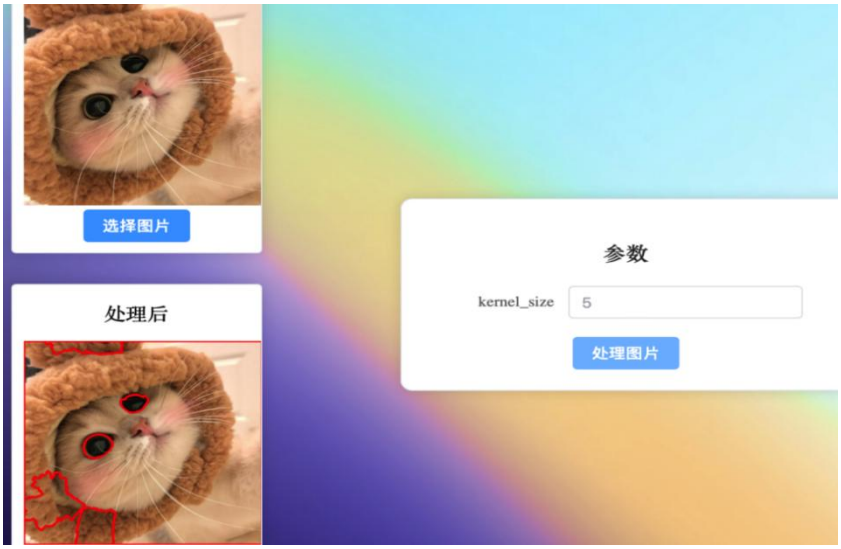

图 5.4 图像分割模块测试

在这次测试中，我们选择的滤波器的大小（kernel_size）参数为默认的 5。可以看到正确显示了阈值处理后的图像。可以看到正确显示了图像分割后的图像。分割边缘由

红色曲线标出。

5）图片目标检测

模型为后端训练后的模型，测试结果如图 5.5：

图 5.5 图片目标检测模块测试

可以看到正确显示了目标检测后的图像。图片用绿色边框和文字标示出了目标位置，目标类别。

6）车牌识别

模型为后端训练后的模型，在前端会显示车牌信息，测试结果如下图 5.6：

图 5.6 车牌识别模块测试

可以看到正确显示了车牌识别后的图像。并显示了正确的车牌号码。

7）视频多目标检测

模型为 yolov8 官方模型测试结果如图 5.7：

图 5.7 视频目标检测模块测试

可以看到正确显示了目标检测后的图像。本功能将返回的数据直接画在原视频上，视频用绿色边框和文字标示出了目标位置，目标类别和置信度。

8）图像均衡化

测试结果如图 5.8：

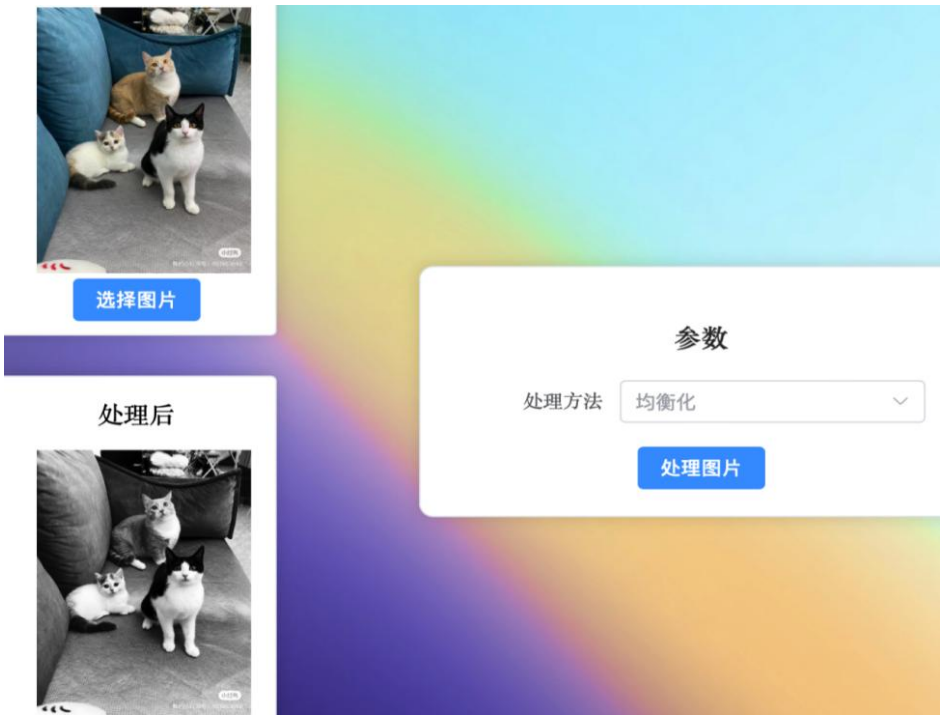

图 5.8 图像均衡化模块测试

可以看到正确显示了图像均衡化后的图像。

9）轮廓检测

用户可以选择算法对应参数，上文有所叙述，测试结果如图 5.9：

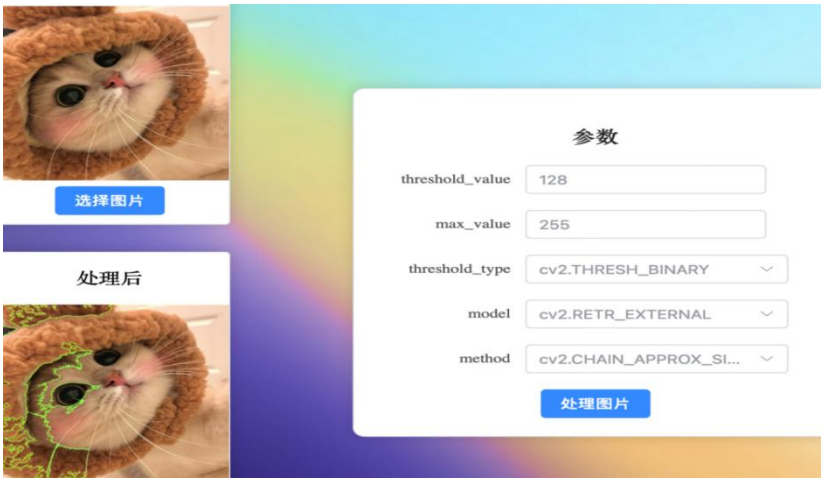

图 5.9 轮廓检测模块测试

在这次测试中，我们选择的起始阈值（threshold_value）参数为默认的 128、最大像素值（max_value）参数为默认的 255、阈值处理类型（threshold_type）参数为默认的 cv2.THRESH_BINARY，为轮廓检测模式（model）参数为默认的 cv2.RETR_EXTERNAL、轮廓近似方法（method）参数为默认的 cv2.CHAIN_APPROX_SIMPLE。

可以看到正确显示了轮廓检测后的图像。轮廓部分由绿色曲线绘制。

10）直线检测

用户可以选择算法对应参数，上文有所叙述，测试结果如图 5.10：

图 5.10 直线检测模块测试

在这次测试中，我们选择的 threshold1 参数为 100、threshold1 参数为默认 200，apertureSize 为默认的 3，并使用精确的 L2 范数进行计算（L2gradient），将参数设为 true，距离参数（rho）为默认的 1、角度参数（theta）为默认的π/180，阈值参数（threshold）

为默认的 100。

可以看到正确显示了直线检测后的图像。直线由红色曲线绘制。

11）角点检测

用户可以选择算法对应参数，上文有所叙述，测试结果如图 5.11：

图 5.11 角点模块测试

在这次测试中，我们选择的邻域大小（blockSize）参数为 8、Sobel 导数算子的孔径大小（ksize）参数为 3，Harris 角点检测参数（k）为默认的 0.04。

可以看到正确显示了角点检测后的图像。角点由红色圈点绘制。

5.1.2 上传图片测试

点击图片上传后，可调用系统文件管理器选取本地图片，并将预处理图片在前端展

示。测试结果如图 5.12 和图 5.13：

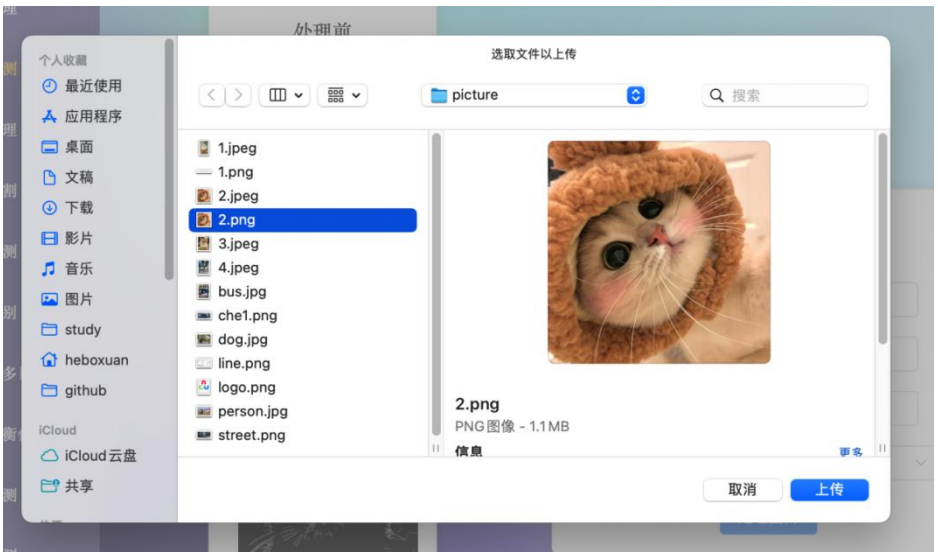

图 5.12 文件管理器

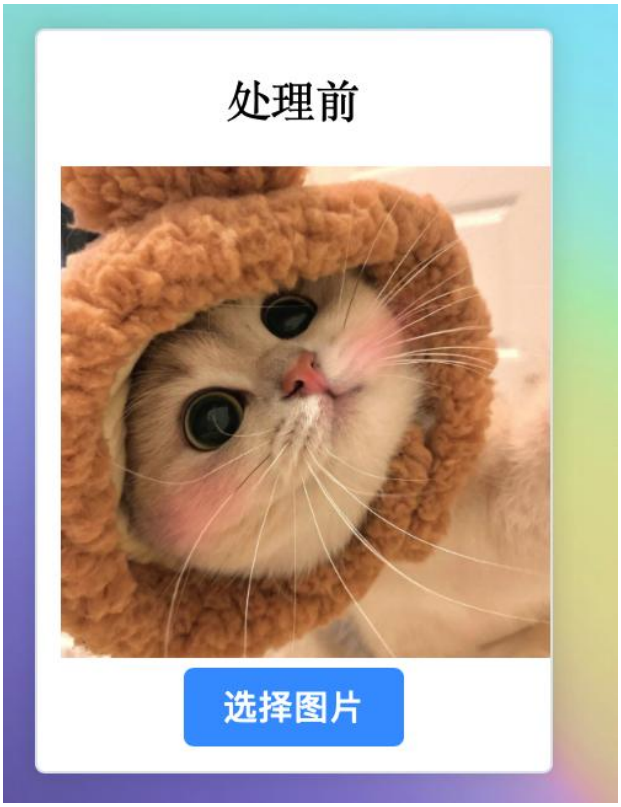

图 5.13 图片上传模块测试

5.1.3 图片处理测试

点击处理，前端可将图像数据以及图像处理参数传至后端，如果接收到返回的处理后图像，则显示处理成功，并将处理后图片展示在前端，未输入参数则按照默认参数处

理，测试结果如图 5.14：

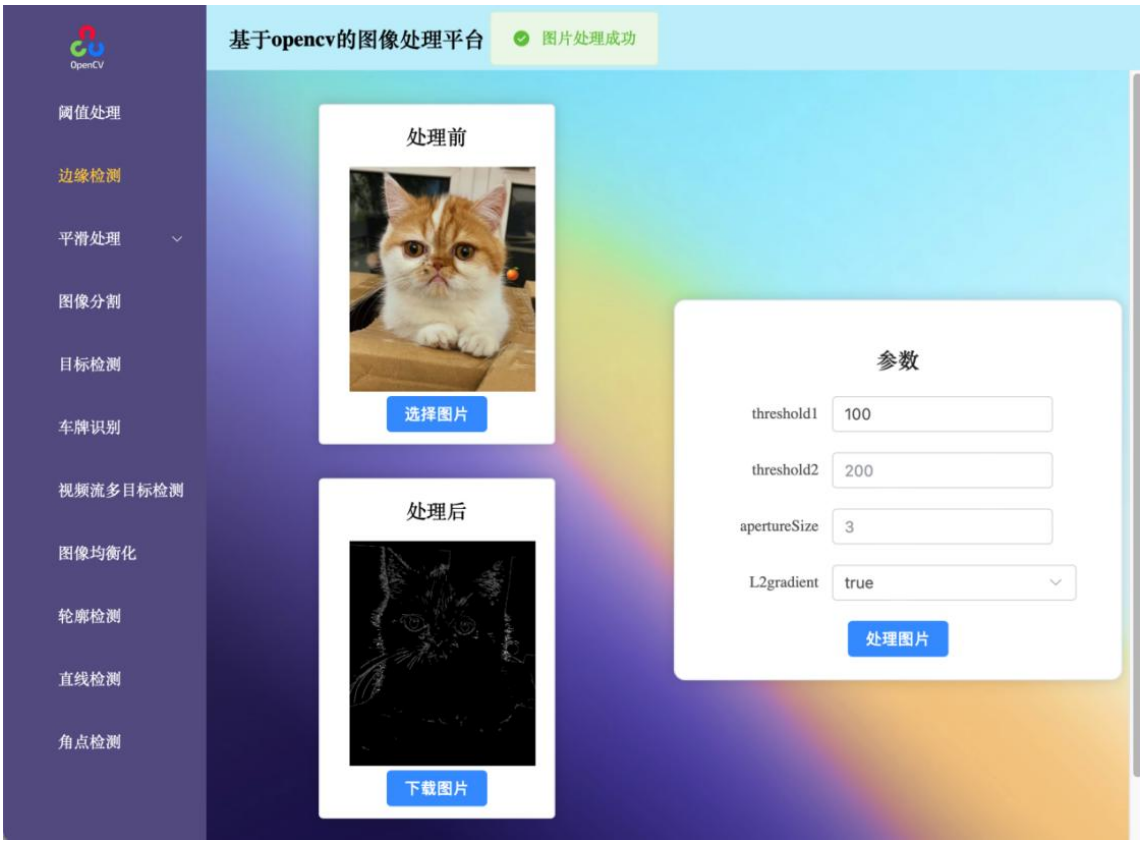

图 5.14 图片处理模块测试

5.1.4 图片下载测试

本平台提供处理后图片的下载功能，可以将图片下载到本地，提示框显示图片下载成功，测试结果如图 5.15：

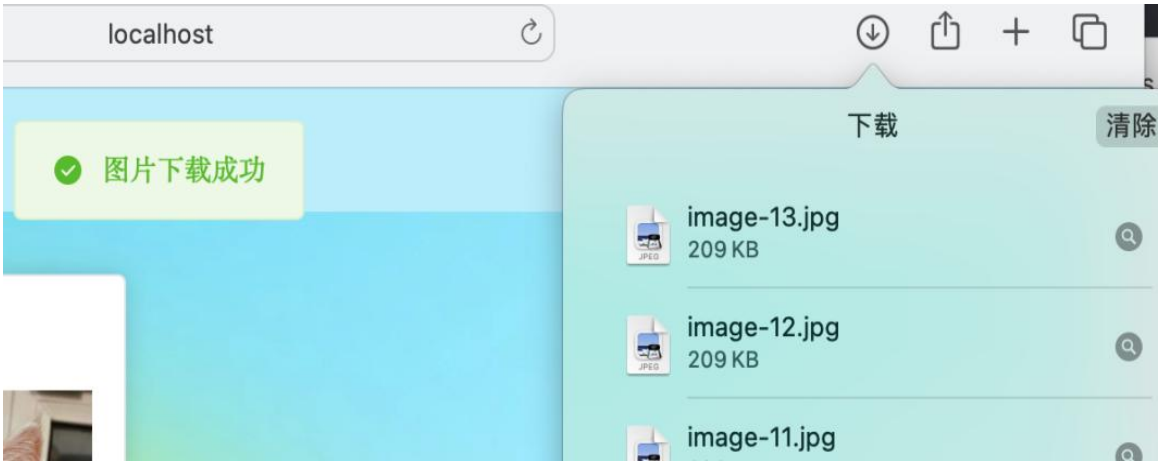

图 5.15 图片下载模块测试

5.1.5 后端接收请求测试

测试后端能否正常接收请求，测试结果如图 5.16：

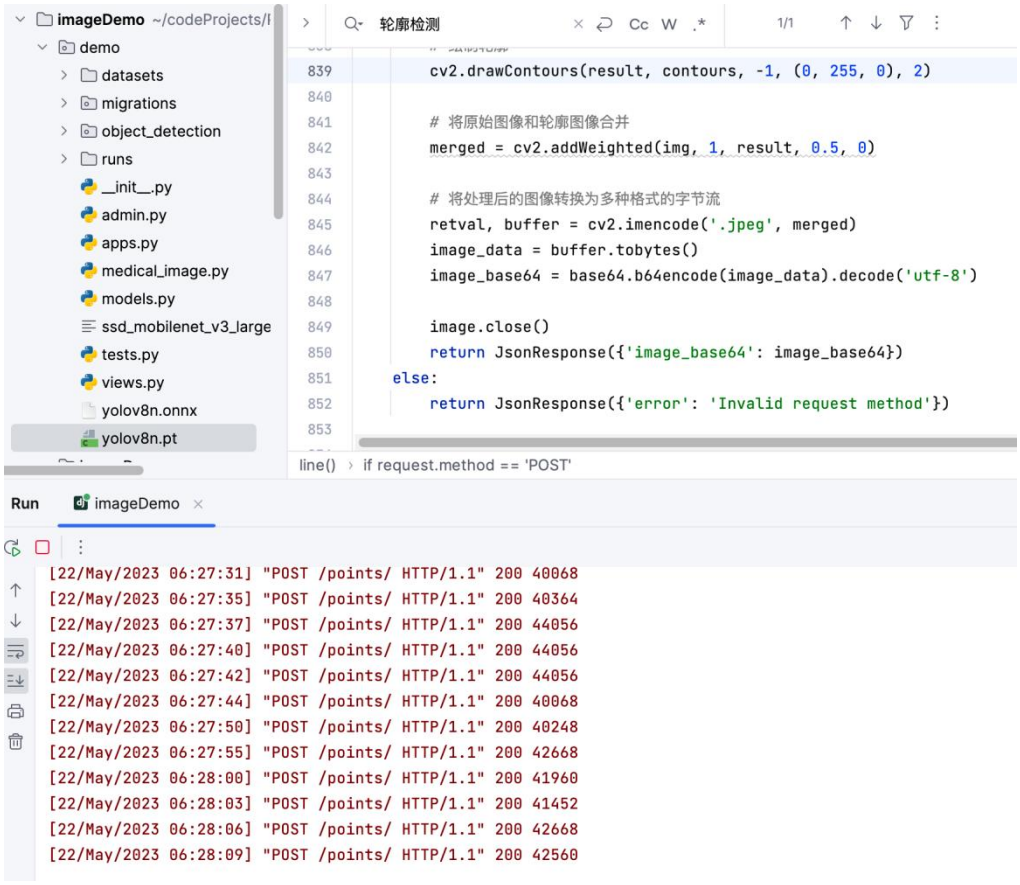

图 5.16 后端接收请求测试

5.1.6 数据限制

图像处理参数应在对应方法的正确范围内。

6 总结

本次毕业设计以 OpenCV 为基础，开发并实现了一个图像处理算法平台，采用了前后端分离的架构。前端使用了 Vue 和 Element Plus 技术栈，后端采用了 Django 框架，而图像处理方面选择了 OpenCV 库。该平台成功实现了多种图像处理功能，包括阈值处理、边缘检测、平滑处理、图像分割、图片目标检测、视频多目标检测和车牌识别。另外，还包括图像均衡化、轮廓检测、直线检测和角点检测等功能。在平滑处理方面，实现了多种滤波方法，包括均值和高斯滤波。

该系统提供了直观易用的可视化界面，将各种基本算法集成在一个平台上，方便用户进行操作，解决了繁琐的图像处理步骤所带来的问题。用户可以根据需求选择不同的方法来满足当前的图像处理需求，体现了设计的灵活性。

在这几个月的学习中，我初步了解了数字图像的相关知识。整个设计过程中，也发现了自己的不足之处：在设计界面时，对 Vue 的掌握不够深入，遇到了许多问题。但通过积极寻求同学们的帮助并上网学习查阅资料，最终解决了这些问题。

本次毕业设计不仅在专业知识上提升了我的能力，还让我明白了如何把握一项任务，从开始到高标准的完成。与此同时，在整个学习和实践过程中，我也提高了查阅文献资料和收集信息的能力，这对我的未来学习将起到积极的作用。

参考文献

[1] 杨福康.计算机图像处理技术的应用与分析[J].电脑知识与技术,2022,18(35):10-13.梁文鹏, 陈磊, 潘伟

[2] 马书群.计算机图像处理技术在 UI 设计中的应用[J].信息记录材料,2020,21(1):95-97.

[3] 王克如. 基于图像识别的作物病虫草害诊断研究[D].中国农业科学院,2005.

[4] 吴园园.数字图像处理关键技术应用与发展[J].计算机产品与流通,2019(03):109.

[5] Li Pei,Wang Hongjuan,Yu Mengbei,Li . Overview of Image Smoothing Algorithms [J]. Journal of Physics: Conference Series,2021,1883(1).

[6] 石炜,仝朝.基于图像去噪的最大均匀平滑法的改进[J].计算机技术与发展,2020,30(11):100-103.

[7] 关雪梅.几种图像平滑处理方法比较研究[J].牡丹江师范学院学报(自然科学版),2016,(04):31-33.

[8] Li Pei,Wang Hongjuan,Yu Mengbei,Li . Overview of Image Smoothing[J]. Journal of Physics: Conference Series,2021,1883(1).

[9] 梁义涛,孟亚敏,朱玲艳等.二维 Otsu 拟合线阈值图像分割方法[J].科学技术与工程,2021,21(09):3689-3697.

[10]Yuyin Wang,Wang Yuyin. Otsu Image Threshold Segmentation Method Based on Seagull Optimization Algorithm[J]. Journal of Physics: Conference Series,2020,1650(3).

[11]姚希.数字图像处理技术及其应用[J].电子技术与软件工程, 2017 (18) :87.

[12]陈之尧.基于 OpenCV-Python 的图像分割技术的设计与应用研究[J].中国新通信,2018,20(19):89.

[13]Kherchaoui S,Houacine A.Face Detection Based on A Model of The Skin Color With Constraints and Template Matching[A].Machine and Web Intelligence(ICMWI),2010 International Conference[C].2010.469-472.

[14]Mehdi Aghagolzadeh,Hamid Soltanian-Zadeh.Multiscale Face Detection:A New Approach to Robust Face Detection[A].2006 IEEEInternational Conference on Fuzzy Systems Sheraton Vancouver Wall Centre Hotel[C].Vancouver,BC,Canada.2006.1229-1234.

[15]Turk M,Pentland A.Eigenfaces for Recognition[J].Journal of Cognitive Neuroscience,

1991,3(1):71-86.

[16]Raphael Sznitman,Bruno Jedynak.Active Testing for Face Detection and Localizatio n[J].IEEETransactions on Pattern Analysis and Machine Intelligence,2010,32(10):1914-1 920.

[17]Mauricio Pamplona Segundo,Luciano,Silva.Automatic Face Segmentation and Facial Landmark Detection in Range Images[J].IEEETransactions on Systems,Man,and Cyber neticsPart B:Cybernetics,2010,40(5):1319-1330.

[18] Lienhart R,Maydt J.An Extended Set of Haarlike Features for Rapid Object Detect ion IEEEICIP 2002,2002,1(1):900-903

致谢

行文至此，写下最后的文字，大学四年即将在六月结束。2019 年秋天充满期待地踏入中北大学，转眼间即将迎来 2023 年夏天的结束。作为中北大学计算机科学与技术学院的一员，我在这里留下了珍贵的青春回忆。四年的时光仿佛昨天，虽然平凡而普通，但我从未停止努力，不断进步和成长。

我要衷心感谢中北大学大数据学院的领导和教职员工们，感谢他们为我们提供了良好的学习环境和资源支持，为我们的专业学习和个人发展打下了坚实的基础。特别感谢物联网专业的各位老师，你们的教诲和辛勤付出让我受益匪浅。您们的严谨教学态度和渊博专业知识激发了我对知识的渴望，推动了我的成长和进步。

我要感谢我的毕业设计指导老师柴锐老师，他全程悉心指导，他的严谨实验态度和渊博的专业知识让我受益匪浅。同时，也要感谢物联网专业的其他老师，他们为我们创造了良好的学习环境，每一位老师都非常负责和严谨，我将铭记他们的教诲，继续努力。

感谢我在大学期间结识的好朋友们，特别感谢宋宝莱同学和柴金铭同学，你们引领我成长，帮助我突破自己的舒适圈，给予我温暖和快乐。感谢办公室的同伴以及学弟学妹们，让我在办公室找到了归属感。感谢物联网 242 班的众多同学，为我的大学生活增添了绚烂的色彩。祝愿我们保持热爱，继续前行，在高处再相聚。

父母的爱是深远的。我要感谢父母二十多年来的照顾和培养，他们让我拥有独立健全的人格。父母的无私付出一直是我前进的动力。我永远铭记他们的养育之恩，将继续努力，成为他们的骄傲。

同时，我也要感谢章惠婷同学。大一时的我稚嫩而幼稚，大二时遇见了她，我们擦出了爱情的火花，这让我第一次体验到了爱情的美好。在与她相处的两年中，她陪伴我成长，帮助我摆脱幼稚，让我更上一层楼。尽管最终未能长久走下去，但她给我带来的改变是巨大的，我的成长也是迅速的。我非常感谢章惠婷同学，在我们共度的时光里。祝愿她未来一切顺利，前程似锦。

以梦为马，不负韶华。感谢一直努力的自己，虽然路途中经历过困惑和挫折，但我从未放弃内心的信念，不断前行，从未停步。每一次经历都成为我人生宝贵的财富。

回忆是我们迫切需要的财富，它们见证了我们成长的足迹。在这四年里，有太多珍贵的回忆和汗水。纵然内心充满不舍，我仍然微笑着与过去告别。

祝福吾校，孕育英才!

祝福吾师，身体健康!

祝福吾友，前程似锦!