**《机器视觉》实验报告**

系 （院）： 计算机与信息学院

专 业： 智能科学与技术

班 级： 智能科23级2班

姓 名： 王晨阳

学 号： 2023217553

指导教师： 吴晶晶

**学年学期:**  2025 ~ 2026 学年 第一学期

2025年12月27日

**实验一 图像滤波**

一、实验内容

本次实验手动实现核心算法完成实验要求。通过编写自定义函数实现图像灰度化，基于手写的二维卷积函数，使用 Sobel 算子对灰度图像滤波，得到 Sobel 算子滤波后的图像；同时尝试使用拉普拉斯卷积核对图像完成滤波操作；通过遍历图像像素的方式手动提取 RGB 三个通道的颜色直方图并完成可视化；最后从 Sobel 算子滤波结果中提取梯度均值、标准差、最大值作为图像纹理特征，并将该特征数组保存至 npy 格式文件中，最终输出所有实验要求的结果。

二、实现方案

实验先通过 OpenCV 读取指定路径下的自主拍摄图像，基于灰度化公式 Y = 0.299R + 0.587G + 0.114B ，完成 RGB 图像到灰度图像的转换；编写二维卷积函数，通过图像边界填充、像素遍历、卷积核逐元素运算的方式实现滤波逻辑，将 Sobel x（实验指定卷积核）、y 方向算子传入该函数完成滤波，再通过梯度融合得到 Sobel 算子滤波图像，同时将拉普拉斯卷积核传入该函数完成额外滤波尝试；编写颜色直方图计算函数，初始化长度为 256 的数组，遍历 RGB 各通道像素统计 0-255 像素值的出现次数，得到颜色直方图数据；从 Sobel 滤波结果中提取梯度的均值、标准差、最大值作为纹理特征，使用 numpy 将特征保存为 npy 格式；最后利用 matplotlib 搭建可视化布局，展示原始图像、灰度图像、各滤波结果图像及颜色直方图，展示实验结果。

三、核心功能实现说明

代码主要由灰度化、二维卷积、颜色直方图计算三部分构成。rgb\_to\_gray函数主要负责灰度化，该函数接收 RGB 图像数组，按照 0.299、0.587、0.114 的权重分别对 R、G、B 三个通道进行加权计算，将计算结果转换为浮点型数组后返回。

conv2d函数是所有滤波操作的核心，先根据卷积核的高和宽计算填充尺寸，对输入图像进行边界填充以避免边缘像素卷积后信息丢失，再通过双层循环遍历图像的每个像素，截取与卷积核尺寸匹配的图像区域，将该区域与卷积核逐元素相乘后求和，得到卷积后的像素值，最终输出与输入图像尺寸一致的滤波结果。color\_histogram函数用于手动计算颜色直方图，初始化一个长度为 256 的整型数组，通过双层循环遍历输入通道的每个像素，将对应像素值位置的数组元素计数加 1，最终返回该通道的直方图统计结果。

主函数中，首先读取指定路径下的自主拍摄图像，判断图像读取是否成功，将 OpenCV 默认的 BGR 格式转换为 RGB 格式；调用rgb\_to\_gray函数得到灰度图像后，定义 Sobel x 方向算子（即实验要求的给定卷积核）和 y 方向算子，将这两个算子分别传入conv2d函数，对灰度图像进行卷积得到 x、y 方向的梯度图，通过计算两个梯度图的平方和开方，得到最终的 Sobel 算子滤波图像；同时定义拉普拉斯卷积核，传入conv2d函数完成滤波，并对结果进行绝对值处理和归一化至 0-255 范围。随后将 RGB 图像的三个通道分别传入color\_histogram函数，得到各通道的颜色直方图数据；基于 Sobel 算子滤波结果计算梯度的均值、标准差、最大值，将这三个值组成数组后通过np.save保存为texture\_feature.npy文件。最后使用 matplotlib展示结果。

四、结果展示

****

图 1 测试输入图片

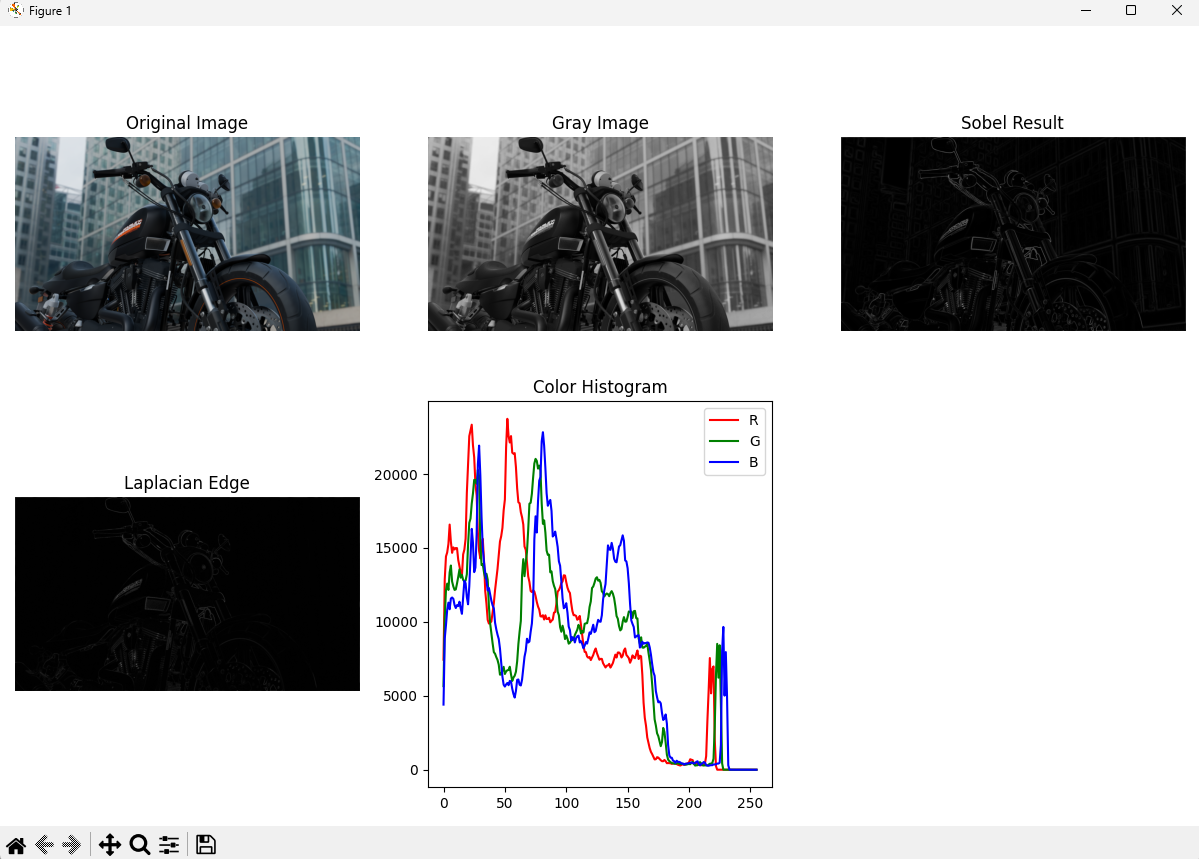
****

图 2 测试结果展示

**实验二 车道线检测**

一、实验内容

本次实验以画有车道线的道路图像（文件路径为 "road.jpg"）作为输入，基于霍夫变换算法完成车道线检测的核心任务。实验全程通过分步骤的图像处理流程实现任务要求：先利用 OpenCV 完成图像预处理，依次执行灰度化、高斯模糊、Canny 边缘检测操作提取图像边缘特征；再根据校园道路的视觉特征自定义梯形感兴趣区域（ROI），过滤掉路边非道路区域的干扰信息，仅保留道路区域的边缘；接着调用概率霍夫变换（HoughLinesP）检测图像中的直线，并通过设置严格的霍夫参数和斜率范围筛选（仅保留 0.4-0.7 和 - 0.7- -0.4 区间的斜率），精准提取符合车道线特征的线段；最后将检测到的车道线以红色线段形式绘制在原始图像上，输出标注了车道线位置的图像结果，同时完成原始图像与检测结果图像的可视化对比展示，并将结果图像保存至本地。

二、实现方案

实验采用分阶段处理的思路完成车道线检测任务。首先读取指定路径下道路图像；随后对彩色图像进行预处理，先转换为灰度图以减少色彩维度带来的计算量，再通过 5×5 的高斯模糊降低图像噪声，消除非必要的细节干扰，接着使用 Canny 边缘检测算法（阈值设为 50 和 150）提取图像中的边缘信息，为后续直线检测提供清晰的特征基础；之后根据校园道路的视觉特点，定义梯形的感兴趣区域，通过掩码操作仅保留道路区域的边缘，排除路边草地、建筑等无关区域的干扰；再调用概率霍夫变换检测直线，通过设置更高的检测阈值、更长的最小线段长度、更小的线段间隙过滤杂线，同时筛选斜率在合理区间内的直线，仅保留符合双向车道线斜率特征的线段；最后将筛选后的车道线绘制在原始图像上，通过图像加权融合得到自然的标注效果，完成结果图像的本地保存，并利用 matplotlib 将原始图像与检测结果图像进行可视化对比，直观呈现车道线检测效果。

三、核心功能实现说明

代码核心由图像预处理、感兴趣区域提取、车道线检测、车道线绘制四部分构成。preprocess\_image函数是图像预处理的核心，该函数接收原始彩色图像，先通过cv2.cvtColor将 BGR 格式的彩色图像转换为灰度图，再调用cv2.GaussianBlur执行 5×5 的高斯模糊操作降低噪声，最后使用cv2.Canny（低阈值 50、高阈值 150）提取图像边缘，返回边缘检测后的单通道图像，为后续直线检测提供清晰的边缘特征。region\_of\_interest函数用于提取道路区域，先获取图像的高和宽，根据校园道路的透视特点定义梯形顶点（左下为宽度 1/4 处、图像底部，左上 / 右上为宽度中点两侧 20 像素、高度 30% 处，右下为宽度 3/4 处、图像底部），创建与边缘图像尺寸一致的全 0 掩码，通过cv2.fillPoly将梯形区域填充为 255，再利用cv2.bitwise\_and将边缘图像与掩码做与运算，仅保留道路区域的边缘信息，过滤路边干扰。

detect\_lane\_lines函数主要负责车道线检测，先获取图像尺寸，调用cv2.HoughLinesP并设置精准的参数（rho=1、theta=np.pi/180、threshold=50、minLineLength=80、maxLineGap=30）检测直线，随后遍历所有检测到的直线，计算每条直线的斜率，排除垂直直线后，筛选出斜率在 - 0.7~-0.4 或 0.4~0.7 区间内的直线，仅保留符合车道线斜率特征的线段。draw\_lane\_lines函数用于绘制车道线，先创建与原始图像尺寸一致的全 0 图像，遍历筛选后的车道线，通过cv2.line绘制红色（0,0,255）、宽度为 3 的线段，再利用cv2.addWeighted将原始图像与车道线图像加权融合（原始图像权重 0.8，车道线图像权重 1），得到标注自然的检测结果图像。

主函数main接收图像路径，读取图像并校验读取状态，依次调用上述函数完成预处理、ROI 提取、车道线检测、车道线绘制，最终完成车道线位置标注的输出要求。

四、结果展示



图 3 测试输入图片



图 4 检测结果图片

**实验三 学号识别**

一、实验内容

本次实验以 MNIST 手写数字数据集为训练集，设计基于卷积神经网络（CNN）的手写数字识别方法，完成自主拍摄的 10 位学号照片的识别任务。实验将本地的学号照片作为输入，首先配置 CPU 计算环境及 batch\_size、学习率、训练轮数等超参数，加载并预处理 MNIST 数据集；随后构建包含两层卷积、两层池化、两层全连接的 CNN 模型，通过 Adam 优化器和交叉熵损失函数完成模型训练与测试，验证模型在 MNIST 测试集上的准确率；核心环节为学号照片的处理，通过灰度化、自适应二值化、形态学去噪、轮廓检测与筛选，将学号照片分割为 10 个独立的数字区域，再将每个数字区域预处理为与 MNIST 数据集格式一致的张量；最后利用训练好的 CNN 模型逐个识别分割后的数字，拼接得到完整的 10 位学号作为输出，同时可视化分割与识别结果，并将训练好的模型保存至本地。

二、实现方案

实验采用 “模型训练 + 图像分割 + 数字识别” 的整体思路完成学号识别任务。首先配置计算环境为 CPU，设定 batch\_size=64、学习率 0.001、训练轮数 5 的超参数，通过 torchvision 加载 MNIST 数据集，利用 ToTensor 和 Normalize 完成数据预处理，将图像转为张量并标准化；接着构建 CNN 模型，通过卷积层提取数字的空间特征，池化层降维，全连接层完成分类，使用交叉熵损失函数计算预测误差，Adam 优化器更新模型参数，分轮次训练模型并在测试集上验证准确率；核心的学号照片处理环节，先将彩色照片转为灰度图，通过自适应二值化增强数字与背景的对比度，形态学开运算去除小噪声，再查找图像轮廓并按面积、宽高比筛选出数字轮廓，按 x 坐标排序保证数字顺序，裁剪每个数字区域并调整为 28×28 的 MNIST 标准尺寸，完成归一化后转为张量；最后将处理后的数字张量逐个输入训练好的 CNN 模型，通过模型推理得到每个数字的识别结果，拼接结果得到完整学号，同时可视化分割后的数字区域与识别结果，保存训练好的模型文件，完成整个学号识别流程。

三、核心功能实现说明

代码核心由环境配置、数据集加载、CNN 模型定义、模型训练测试、学号图片分割、学号识别及实际调用七部分构成。首先完成环境配置，指定 CPU 为计算设备，设定 batch\_size、learning\_rate、epochs 等超参数；通过 transforms.Compose 组合 ToTensor 和 Normalize 操作，加载 MNIST 训练集与测试集，并用 DataLoader 构建数据加载器，实现数据的批量读取与打乱。

CNN 模型定义在MNIST\_CNN类中，该类包含两层卷积层（conv1 将 1 通道转为 32 通道，conv2 将 32 通道转为 64 通道，均使用 3×3 卷积核并填充 1 像素）、一层 MaxPool2d 池化层（2×2 核）、两层全连接层（fc1 将 64×7×7 的特征展平后映射为 128 维，fc2 映射为 10 维对应 0-9 数字分类），同时引入 ReLU 激活函数增加非线性，Dropout 层（概率 0.25）防止过拟合；forward 函数实现模型前向传播，依次执行卷积、激活、池化、展平、全连接等操作，输出分类结果。

train\_model函数实现模型训练，将模型设为训练模式，遍历训练集批次数据，将图像和标签送入 CPU，通过模型得到预测输出，计算损失值后反向传播更新参数，每 100 步打印一次损失值；test\_model函数将模型设为评估模式，在无梯度计算的环境下遍历测试集，统计预测正确的样本数，计算并返回模型测试准确率。

split\_student\_id\_image函数是学号识别的核心，实现学号照片的分割与预处理：先读取照片转为灰度图，通过自适应二值化将数字转为黑色、背景转为白色，再用 2×2 的核执行形态学开运算去除小噪声；调用 findContours 查找外部轮廓，按面积≥50、宽高比 0.2~2 的条件筛选数字轮廓，按 x 坐标排序保证学号顺序；对每个轮廓裁剪数字区域（上下左右各扩 2 像素避免切边），调整为 28×28 尺寸，完成与 MNIST 一致的归一化后转为张量，返回数字张量列表和原始裁剪图列表，若检测到的轮廓数非 10 则可视化轮廓辅助排查问题。

recognize\_student\_id函数实现数字识别，将模型设为评估模式，调用 split\_student\_id\_image 得到数字张量，逐个输入模型推理，通过 torch.max 获取预测结果，拼接得到学号；同时创建画布可视化每个数字的裁剪图与识别结果，返回最终学号。

四、结果展示

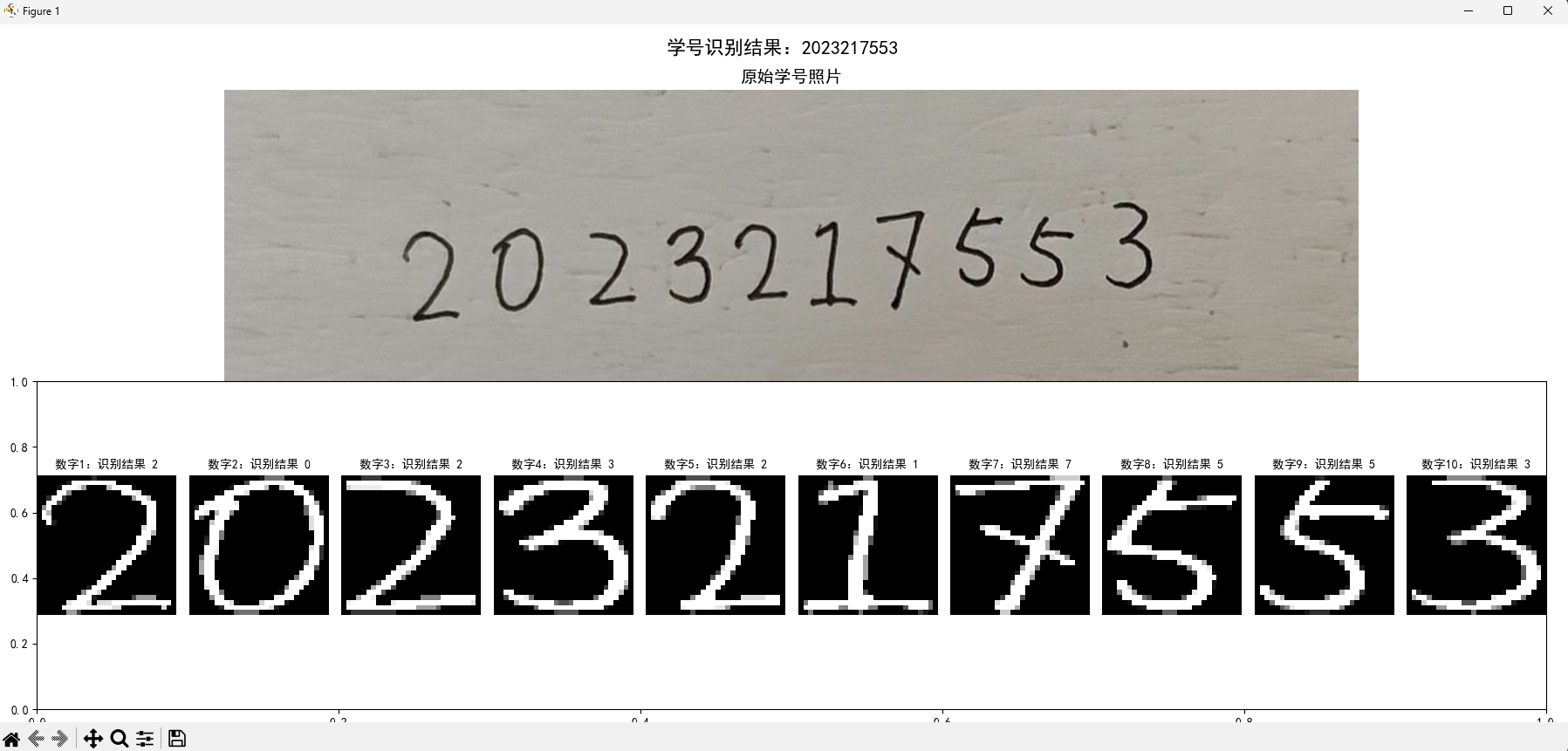


图 5 识别结果

**实验四 校园共享单车检测**

一、实验内容

本次实验以基于 COCO 数据集预训练的 YOLOv8s 模型为核心，完成校园场景下哈罗共享单车的检测任务。实验将本地路径的校园共享单车照片作为输入，在 CPU 环境下加载 YOLOv8s 预训练模型，先指定仅检测 COCO 数据集中类别 ID 为 1 的自行车目标，完成图像中自行车的定位；再针对哈罗单车专属的蓝色特征，设计双区间 HSV 色彩范围（主色宝蓝、辅助色浅青蓝），判断检测到的自行车是否为哈罗品牌，最终输出哈罗单车的位置（以检测框 [x1, y1, x2, y2] 坐标呈现）；同时可视化标注哈罗单车的检测框（蓝色框 + 专属标签），展示检测结果并将标注后的图像保存至指定目录。

二、实现方案

首先加载基于 COCO 数据集预训练的 YOLOv8s 模型，设定检测参数（CPU 运行、置信度 0.25、IOU0.45），指定仅检测自行车类别，对输入的共享单车照片完成自行车目标的定位，输出所有自行车的检测框；接着针对哈罗单车的蓝色视觉特征，设计双区间 HSV 色彩范围（主色宝蓝：H105-125、S70-255、V40-255；辅助色浅青蓝：H95-110、S50-255、V60-255），裁剪检测框内的自行车区域并轻微缩小 5%（排除背景干扰），将区域图像转换为 HSV 色彩空间以减少光照影响，生成双区间蓝色掩码并合并，计算蓝色像素占比；设定蓝色占比阈值（默认 0.55，实验中调整为 0.35），判断检测框内的自行车是否为哈罗单车；最后对判定为哈罗单车的目标绘制专属蓝色检测框与标签，可视化展示检测结果，将标注后的图像保存至指定目录，输出哈罗单车的位置信息，完成整个检测流程。

三、核心功能实现说明

代码核心由哈罗单车颜色判断函数、检测主函数及执行调用三部分构成。is\_hello\_bicycle函数是品牌分类的核心，该函数接收原始图像与自行车检测框坐标，先将检测框缩小 5%（减少背景干扰）并做边界保护（避免坐标越界），裁剪出自行车主体区域；若裁剪后无有效像素则直接返回 False；随后将裁剪区域转换为 HSV 色彩空间，定义哈罗单车专属的双区间蓝色 HSV 范围，生成两个区间的掩码并通过位或运算合并，统计蓝色像素数量并计算占比；对比设定的蓝色阈值，返回是否为哈罗单车的布尔值及蓝色像素占比，同时打印调试日志展示阈值与实际占比。

detect\_hello\_bicycle函数主要负责检测，先校验图片文件是否存在、是否可读，读取原始图像后调用 YOLOv8s 模型，指定仅检测自行车类别、CPU 运行、置信度 0.25、IOU0.45 等参数，得到图像中所有自行车的检测结果；遍历每个自行车检测框，调用is\_hello\_bicycle函数判断是否为哈罗单车，若为哈罗单车则记录其检测框、置信度、蓝色占比信息，同时在标注图像上绘制蓝色检测框（边框宽度 3）与包含置信度、蓝色占比的专属标签；完成所有检测框处理后，通过 OpenCV 展示标注后的检测结果，按参数要求保存结果图像至指定目录，最终返回哈罗单车的检测结果列表。

执行调用部分指定共享单车照片的本地路径，调用detect\_hello\_bicycle函数，设置保存检测结果、蓝色占比阈值为 0.35，触发整个检测流程，输出哈罗单车的位置信息（检测框坐标），完成校园哈罗共享单车的定位与识别。

四、结果展示

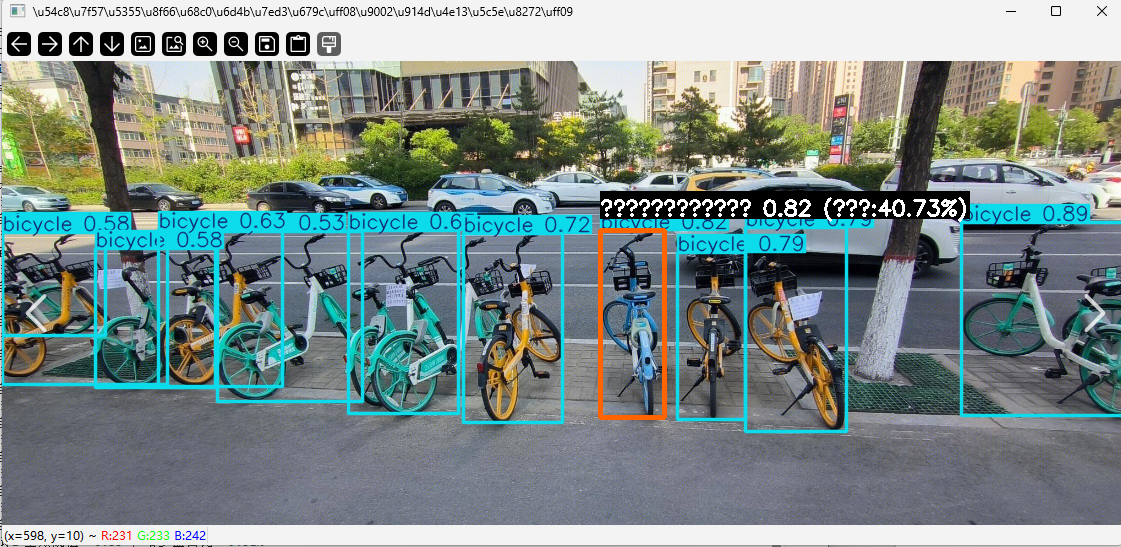


图 6 最终单独检测出哈啰单车

**实验加分项实现说明**

一、创建自己名字命名的代码环境

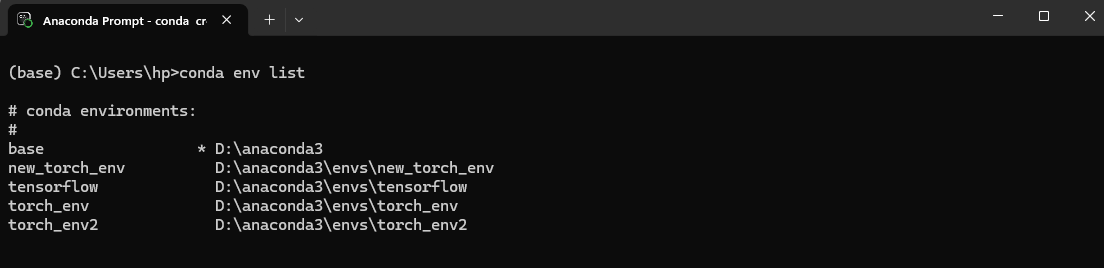


图 7 先查看当前已有的环境

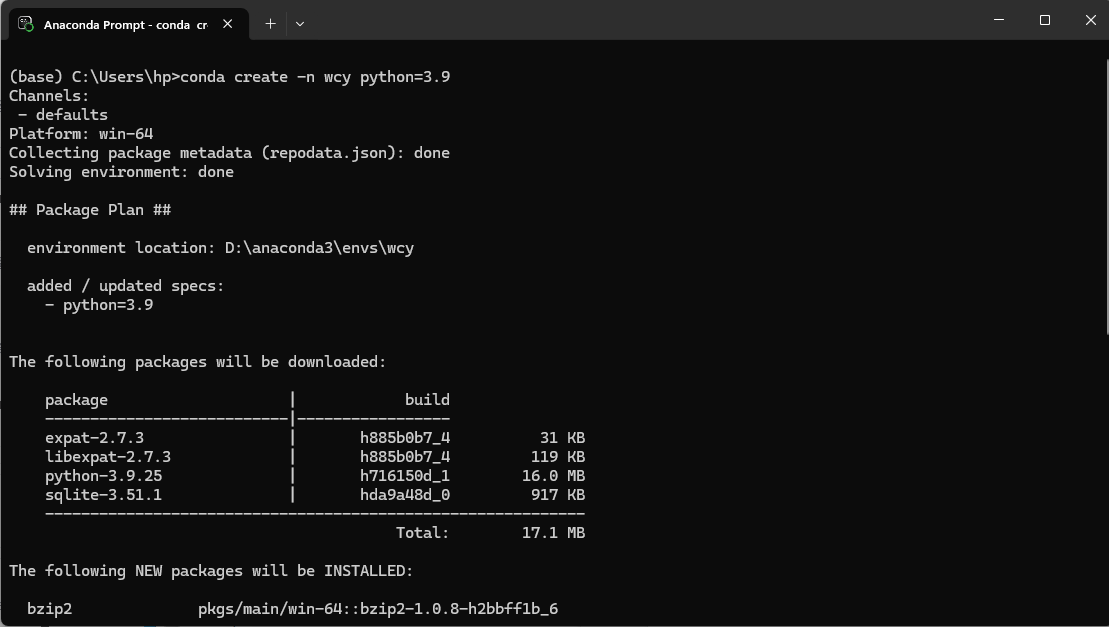


图 8 创建名字缩写wcy的环境

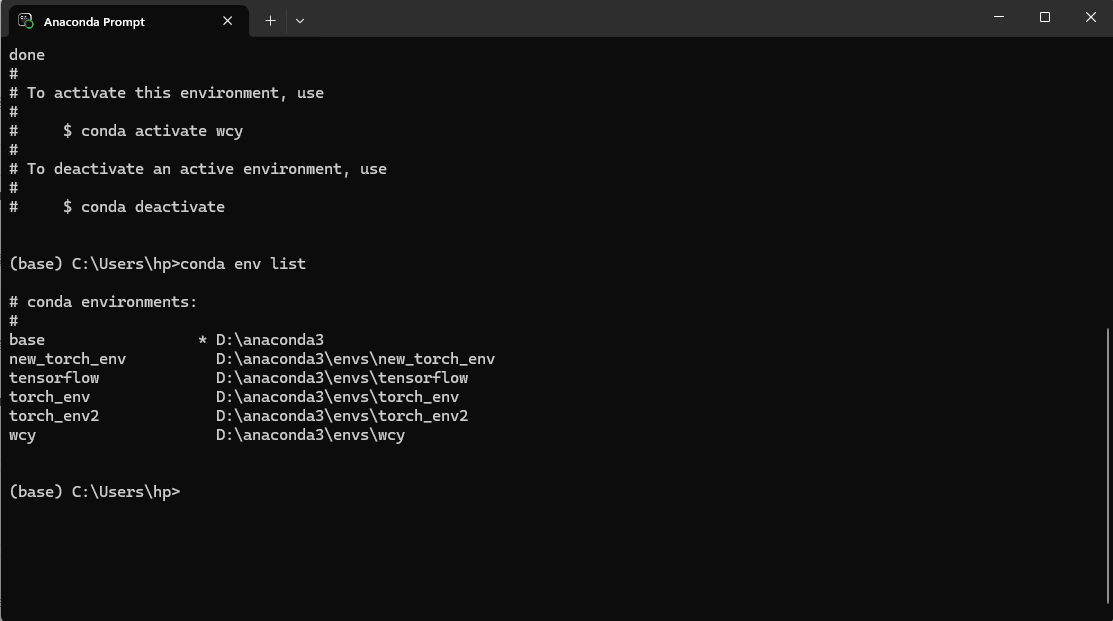


图 9 创建成功

1. git提交作业

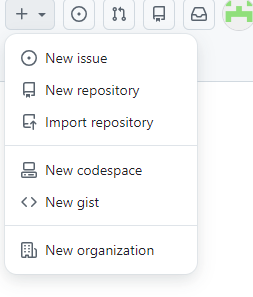


图 10 创建新仓库

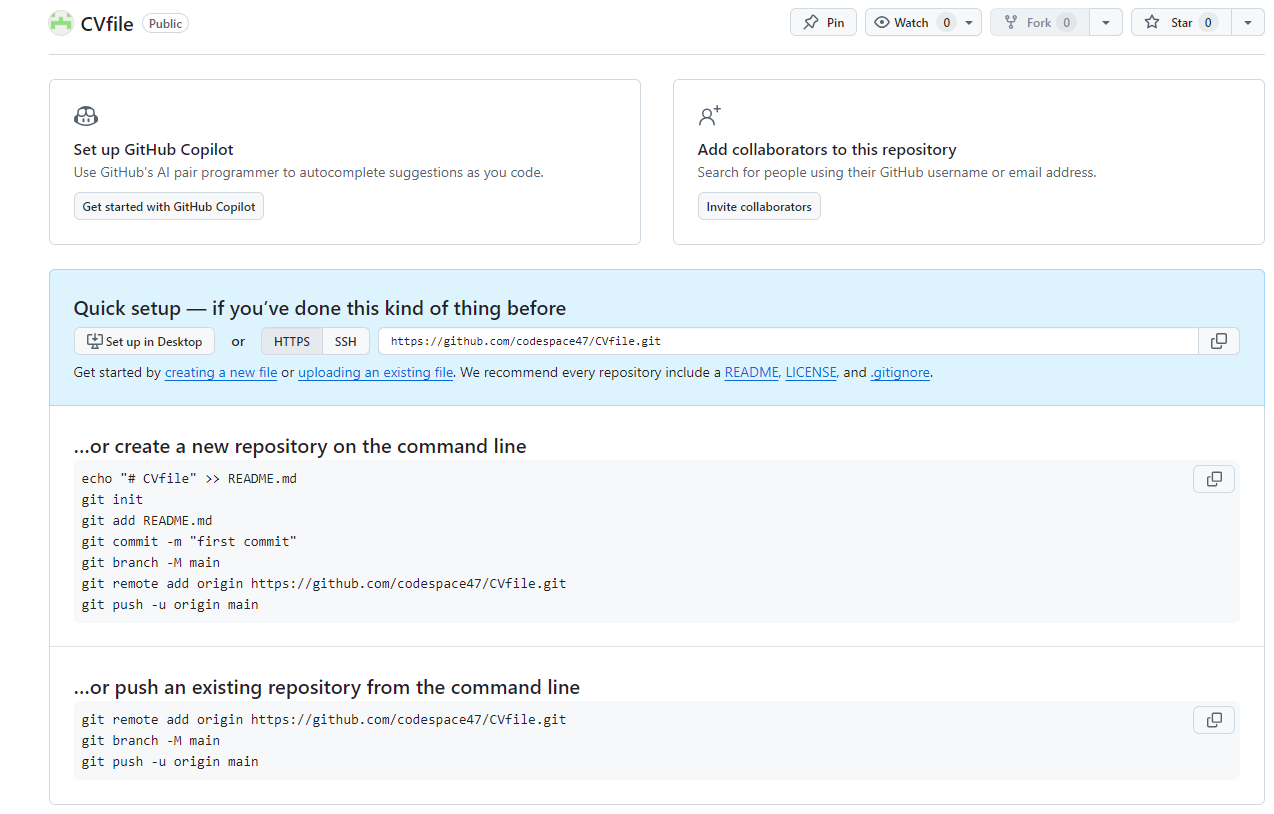
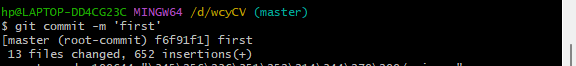
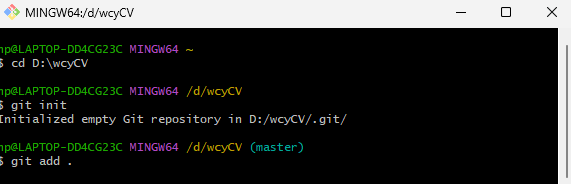


图 11 新仓库界面



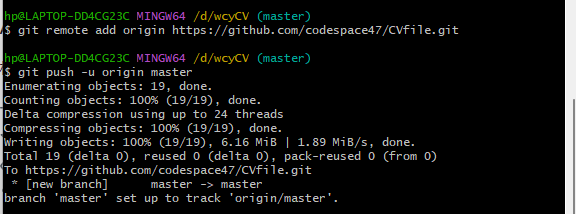


图 12 通过git提交