# R2-Gaussian 一键环境配置脚本

以下是整合所有步骤的**一键执行脚本**，包含环境创建、依赖安装、变量配置、子模块编译全流程，直接复制到终端运行即可（建议在项目根目录执行）。

## 1. 一键配置脚本（完整版）

|  |
| --- |
| #!/bin/bash  set -e # 脚本出错时自动退出，避免继续执行  ###########################################################################  # 步骤1：定义常量与环境检查  ###########################################################################  ENV\_NAME="r2\_gaussian" # conda环境名  PYTHON\_VERSION="3.9" # 项目兼容Python版本  CUDA\_VERSION="11.6" # 统一CUDA版本  GCC\_VERSION="8.5.0" # 兼容CUDA的GCC版本  # 检查Anaconda是否安装  if ! command -v conda &> /dev/null; then  echo "错误：未检测到Anaconda，请先安装Anaconda并配置环境变量！"  exit 1  fi  # 检查项目根目录是否存在子模块文件夹  if [ ! -d "r2\_gaussian/submodules" ]; then  echo "错误：未找到r2\_gaussian/submodules目录，请先克隆项目并确保子模块已拉取！"  echo "提示：克隆项目命令：git clone https://github.com/Ruyi-Zha/r2\_gaussian.git"  exit 1  fi  ###########################################################################  # 步骤2：创建并激活conda环境  ###########################################################################  echo -e "\n=== 1/6 创建conda环境 $ENV\_NAME ==="  # 检查环境是否已存在，存在则先删除（避免旧环境干扰）  if conda info --envs | grep -q "$ENV\_NAME"; then  echo "环境 $ENV\_NAME 已存在，正在删除旧环境..."  conda remove --name "$ENV\_NAME" --all -y  fi  # 创建新环境  conda create -n "$ENV\_NAME" python="$PYTHON\_VERSION" -y  # 激活环境（临时生效，后续步骤依赖此环境）  source "$(conda info --base)/etc/profile.d/conda.sh"  conda activate "$ENV\_NAME"  ###########################################################################  # 步骤3：安装CUDA相关依赖（编译器+开发包+兼容GCC）  ###########################################################################  echo -e "\n=== 2/6 安装CUDA $CUDA\_VERSION 与编译器 ==="  # 安装CUDA核心依赖（含nvcc编译器、头文件、运行时库）  conda install \  cudatoolkit-dev="$CUDA\_VERSION" \  cuda-nvcc="$CUDA\_VERSION" \  -c conda-forge \  -c nvidia/label/cuda-"$CUDA\_VERSION".0 \  -y  # 安装兼容的GCC/G++（避免版本过高导致CUDA编译失败）  conda install \  -c conda-forge \  gcc="$GCC\_VERSION" \  gxx="$GCC\_VERSION" \  -y  ###########################################################################  # 步骤4：配置环境变量（永久生效+临时生效）  ###########################################################################  echo -e "\n=== 3/6 配置环境变量 ==="  # 获取conda环境路径  CONDA\_ENV\_PATH=$(conda info --envs | grep "$ENV\_NAME" | awk '{print $2}')  # 1. 临时生效（当前终端，用于后续编译）  export CUDA\_HOME="$CONDA\_ENV\_PATH"  export CUDACXX="$CONDA\_ENV\_PATH/bin/nvcc"  export C\_INCLUDE\_PATH="$CONDA\_ENV\_PATH/include:$C\_INCLUDE\_PATH"  export CPLUS\_INCLUDE\_PATH="$CONDA\_ENV\_PATH/include:$CPLUS\_INCLUDE\_PATH"  export LIBRARY\_PATH="$CONDA\_ENV\_PATH/lib:$LIBRARY\_PATH"  export LD\_LIBRARY\_PATH="$CONDA\_ENV\_PATH/lib:$LD\_LIBRARY\_PATH"  export CC="$CONDA\_ENV\_PATH/bin/gcc"  export CXX="$CONDA\_ENV\_PATH/bin/g++"  # 2. 永久生效（写入bashrc，所有终端生效）  echo "  # R2-Gaussian 环境变量配置（自动生成）  export CUDA\_HOME=$CONDA\_ENV\_PATH  export CUDACXX=$CONDA\_ENV\_PATH/bin/nvcc  export C\_INCLUDE\_PATH=$CONDA\_ENV\_PATH/include:\$C\_INCLUDE\_PATH  export CPLUS\_INCLUDE\_PATH=$CONDA\_ENV\_PATH/include:\$CPLUS\_INCLUDE\_PATH  export LIBRARY\_PATH=$CONDA\_ENV\_PATH/lib:\$LIBRARY\_PATH  export LD\_LIBRARY\_PATH=$CONDA\_ENV\_PATH/lib:\$LD\_LIBRARY\_PATH  export CC=$CONDA\_ENV\_PATH/bin/gcc  export CXX=$CONDA\_ENV\_PATH/bin/g++  " >> ~/.bashrc  # 生效bashrc配置  source ~/.bashrc  # 重新激活环境（确保变量完全生效）  conda activate "$ENV\_NAME"  ###########################################################################  # 步骤5：安装PyTorch及项目依赖  ###########################################################################  echo -e "\n=== 4/6 安装PyTorch与项目依赖 ==="  # 安装PyTorch 2.1.2+cu118（与CUDA 11.6兼容，minor版本差异不影响）  pip install torch==2.1.2+cu118 torchvision==0.16.2+cu118 \  --extra-index-url https://download.pytorch.org/whl/cu118  # 安装项目其他依赖（从requirements.txt，若不存在则跳过）  if [ -f "requirements.txt" ]; then  pip install -r requirements.txt  else  echo "警告：未找到requirements.txt，跳过项目依赖安装，请手动安装所需依赖！"  fi  ###########################################################################  # 步骤6：编译子模块（simple-knn + xray-gaussian-rasterization-voxelization）  ###########################################################################  echo -e "\n=== 5/6 编译 simple-knn 子模块 ==="  # 进入simple-knn目录  cd r2\_gaussian/submodules/simple-knn  # 清理旧编译缓存（避免残留文件干扰）  rm -rf build/ dist/ \*.so simple\_knn.egg-info/  # 编译安装（--no-build-isolation 确保torch被识别）  pip install -e . --no-build-isolation --force-reinstall  # 验证编译结果  if python -c "import simple\_knn" &> /dev/null; then  echo "✅ simple-knn 编译成功！"  else  echo "❌ simple-knn 编译失败，请检查日志！"  exit 1  fi  # 返回项目根目录  cd ../../../  echo -e "\n=== 6/6 编译 xray-gaussian-rasterization-voxelization 子模块 ==="  # 编译安装xray子模块  pip install -e r2\_gaussian/submodules/xray-gaussian-rasterization-voxelization --no-build-isolation  # 验证编译结果  if python -c "import xray\_gaussian\_rasterization\_voxelization" &> /dev/null; then  echo "✅ xray-gaussian-rasterization-voxelization 编译成功！"  else  echo "❌ xray-gaussian-rasterization-voxelization 编译失败，请检查日志！"  exit 1  fi  ###########################################################################  # 步骤7：安装TIGRE（可选，根据项目需求）  ###########################################################################  echo -e "\n=== 可选步骤：安装 TIGRE ==="  if [ -d "TIGRE-2.3/Python" ]; then  cd TIGRE-2.3/Python  pip install . --no-build-isolation  cd ../../  echo "✅ TIGRE 安装成功！"  else  echo "ℹ️ 未找到 TIGRE-2.3/Python 目录，跳过TIGRE安装"  fi  ###########################################################################  # 最终验证与提示  ###########################################################################  echo -e "\n========================================"  echo "🎉 R2-Gaussian 环境配置完成！"  echo "========================================"  echo "当前环境信息："  echo "1. Conda环境：$ENV\_NAME"  echo "2. Python版本：$(python --version | awk '{print $2}')"  echo "3. CUDA版本（nvcc）：$(nvcc -V | grep "release" | awk '{print $6}' | cut -d',' -f1)"  echo "4. GCC版本：$(gcc --version | head -1 | awk '{print $4}')"  echo "5. PyTorch CUDA可用性：$(python -c "import torch; print('可用' if torch.cuda.is\_available() else '不可用')")"  echo -e "\n使用方法："  echo "1. 激活环境：conda activate $ENV\_NAME"  echo "2. 运行项目：根据项目文档执行启动命令（如 python train.py）"  echo "========================================" |

## 2. 脚本使用说明

### 2.1 执行步骤

1. **保存脚本**：将上述代码复制到文本文件，命名为 setup\_r2\_gaussian.sh
2. **赋予执行权限**：

|  |
| --- |
| chmod +x setup\_r2\_gaussian.sh |

1. **运行脚本**（建议在项目根目录执行）：

|  |
| --- |
| ./setup\_r2\_gaussian.sh |

### 2.2 关键说明

* **自动处理冲突**：脚本会先删除已存在的 r2\_gaussian 环境，避免旧依赖干扰
* **错误终止**：通过 set -e 确保出错时自动停止，便于定位问题
* **验证环节**：每个核心步骤后都有验证（如 CUDA 版本、子模块编译），避免 “隐形失败”
* **可选步骤**：TIGRE 安装为可选，若项目不需要可删除对应代码块

### 2.3 常见问题处理

若脚本执行中报错，优先检查以下点：

1. **网络问题**：Anaconda/pip 下载超时 → 配置国内源（如 conda 添加清华源、pip 使用 -i https://pypi.tuna.tsinghua.edu.cn/simple）
2. **子模块缺失**：提示 r2\_gaussian/submodules 不存在 → 先执行 git clone https://github.com/Ruyi-Zha/r2\_gaussian.git 克隆项目
3. **权限问题**：写入 ~/.bashrc 权限不足 → 执行 sudo chmod 644 ~/.bashrc 赋予写入权限

## 3. 脚本执行后验证

脚本执行完成后，可手动执行以下命令确认环境正确性：

|  |
| --- |
| # 激活环境  conda activate r2\_gaussian  # 1. 验证CUDA版本  nvcc -V # 应输出 11.6  # 2. 验证GCC版本  gcc --version # 应输出 8.5.0  # 3. 验证PyTorch CUDA  python -c "import torch; print(torch.version.cuda); print(torch.cuda.is\_available())" # 输出 11.8 + True  # 4. 验证子模块  python -c "import simple\_knn; import xray\_gaussian\_rasterization\_voxelization; print('所有子模块正常加载！')" |

若所有验证均通过，即可按照项目文档运行 R2-Gaussian 相关代码（如训练、推理）。

（注：文档部分内容可能由 AI 生成）