Imaging：节点式图像处理软件

北京大学 2024 春 · 程序设计实习 · Qt 大作业报告

重生之我编程填空都队

沈睿弘 陈睿哲 高美琚

一. 程序功能介绍

1.1 概述

本项目开发了一个图像处理软件 Imaging，使用户能通过增减、移动、连接具有不同功能的函数节点对图像进行各种操作与调整，从而实现图像的调色、美化和再创作。

1.2 图像编辑功能

Imaging 的图像编辑功能通过节点树的连接方式实现，（有别于 Photoshop 等层级式图像处理软件，在复杂应用时采用节点树更直观且易于修改。）在节点树构建完成并触发运行后，软件读取输入节点的数据，按照节点连接顺序逐一调用相应节点的函数进行图像编辑，直至到达输出节点进行输出。一个简单的节点树如图 1-1 所示。

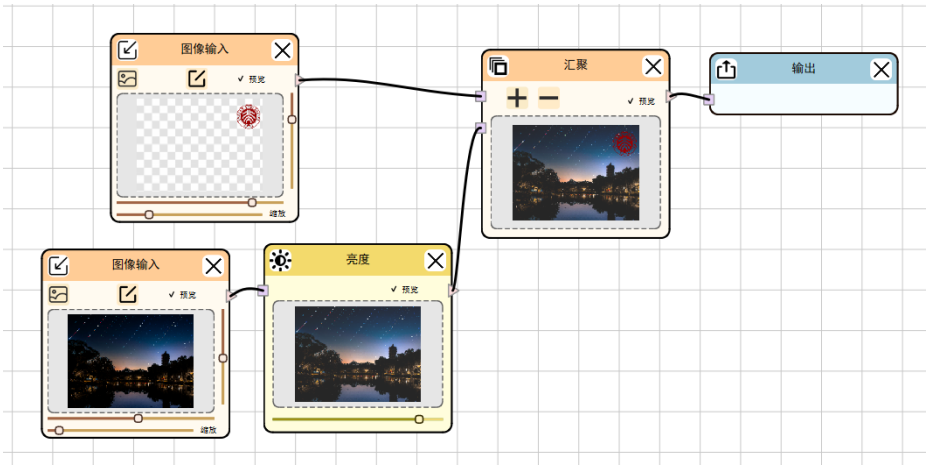

图 1-1. 节点树示意

所有节点均支持图片预览选框、持添加多输出端口、右键快捷菜单（删除节点、复制输出）功能。

1.2.1 输入节点与汇聚节点

输入节点（InputNode）接受用户提供的原始图像，作为整个处理流程的起点，并且执行一些基本的预处理工作。节点 UI 上的三个滑块对应移动和缩放操作；此外，节点还可以唤出预处理窗口支持裁剪、AI 抠图、旋转操作和更精确的移动与缩放操作。

图 1-2. 输入窗口 UI

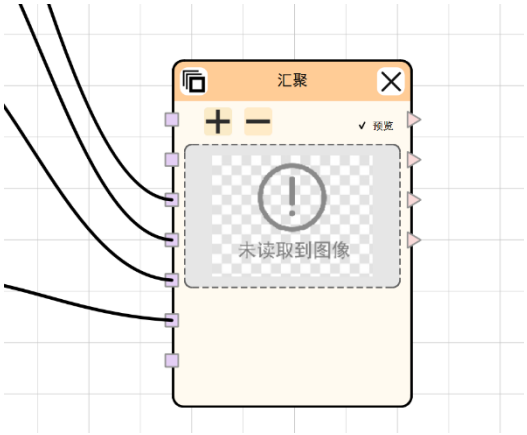

图 1-3.汇聚窗口 UI

汇聚节点（AddNode）支持添加多个图片输入端，并支持图层覆盖顺序的调整。节点支持添加和删除输入端口。UI 边界会根据添加的端口数量进行自动调整。

1.2.2 调整节点

（1）基础调整节点（EditNode）

基础调整节点提供移动、裁剪、缩放、旋转功能，支持唤出编辑窗口进行更精确的操作。

（2）对比度节点（ContrastNode）、亮度节点（BrightnessNode）、饱和度节点（SaturationNode）、色调节点（HueNode）、色温节点（ColorTemperatureNode）

以上五个节点的使用方式相似。节点接受一个输入图像，通过拖动节点 UI 上的滑块进行相应属性参数的调整。

（3）滤镜节点（FilterNode）

滤镜节点接受一个输入图像，还需要读取一个滤镜 LUT 文件，通过拖动滑块进行滤镜施加程度的调整。

1.2.3 AI 节点

AI 节点提供基于 AI 模型的图像处理、生成功能。Imaging 通过调用相应 API 实现了两种 AI 功能，其中，AI 文生图节点（AIword2imgNode）接受一段文字输入作为 prompt 并生成一张图像，AI 图生图节点（AIimg2imgNode）接受一个输入图像和一段文字 prompt，按照 prompt 对输入图像进行相应修改。

1.2.4 输出节点

输出节点（OutputNode）将处理后的图像显示在窗口右侧的预览区域内，支持用户导

出图片文件。

1.3 用户交互功能

除了主要的节点树编辑功能之外，主窗口还提供一些用户交互功能。主窗口界面如图 1-2 所示。

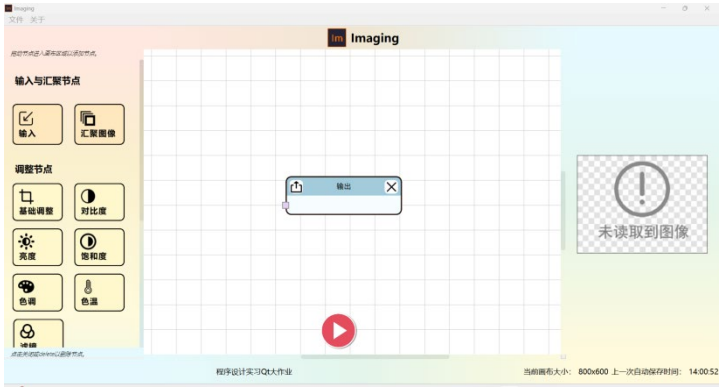

图 1-4. 主窗口 UI

1.3.1 菜单栏

菜单栏提供的功能如表 1-1 所示。

文件	新建、打开、保存项目	新建、读取或保存 ima 文件
	保存图像	保存输出节点接受的图像
	更改画布大小	更改输出图片的大小
	撤销 / Ctrl+Z 快捷键	撤销连线、新建节点操作
	退出	退出软件
关于	关于	查看软件介绍
	帮助	查看演示视频

表 1-1. 菜单栏功能

1.3.2 其他

1.3.2.1 主窗口左侧是节点选择窗口（NodeChoice），支持拖动添加节点。

1.3.2.2 主窗口右侧是图像预览窗口，支持滚轮缩放和双击放大查看。支持双击弹出新窗口进行全尺寸输出图片预览。

1.3.2.3 主窗口中央是编辑区域（MainScene/MainView），用于放置节点和绘制连接线，支持滚轮缩放和拖动。

1.3.2.4 主窗口下侧是信息窗口，显示当前画布大小和上一次自动保存时间，显示并支持修改文件名。

1.3.2.5 编辑区域下部有悬浮的运行按钮，支持手动更新输出图像。

1.4 其他

Imaging 在编辑过程中会自动保存节点树文件，并每隔一定时间自动更新输出图像。

二. 项目细节

2.1 主窗口

2.1.1 编辑区域

编辑区域由 MainScene 和 MainView 管理，二者分别是 QGraphicsScene 和 QGraphicsView 的派生类，因此在接受鼠标事件等时能直接得到作用的对象部件。由 QGraphicsScene 负责管理所有 QGraphicsItem（包括节点 Node、连接线 Connectline、节点端口 Terminal）。当执行节点树时，首先遍历所有节点，执行节点的 execute 函数，完成节点处理，此后遍历所有连接线，取出上一节点的输出作为下一节点的输入，外部接受 Output 节点的唯一输出，显示在右侧预览窗口。（注：我们并未做节点树解析，故未必保证在单刷新步内完成所有节点的结果刷新，但由于定时器事件，在至多 n=节点数量 次刷新后，一定能取得正确结果。由于 OpenCV 的高效性，该方案是可接受的。），编辑区域接受来自节点选择窗口的拖拽放置事件，接受鼠标滚轮事件进行缩放，接受鼠标按下和移动事件以选中对象、拖动节点和绘制连接线。

2.1.2 一些停靠窗口部件

节点选择窗口是 QWidget 的派生类，使用 Qt Designer 设计，其中的节点选项监测鼠标按下事件，并生成一个拖拽事件作为响应，当拖拽到编辑区域并松开鼠标时在相应位置生成一个相应节点。

图像预览窗口涉及 ImageWidget 类的实现，这是 QWidget 的派生类，重写了鼠标事件和滚轮事件，用于显示图片并支持缩放和双击唤出显示全尺寸图片预览。

2.2 节点实现细节

所有节点（Node）是 QGraphicsItem 的派生类，Imaging 共实现了 12 个节点，结点的外观如图 2-1 所示，节点的派生关系和一些相关控件的包含关系如图 2-2 所示。

图 2-1. 节点外观

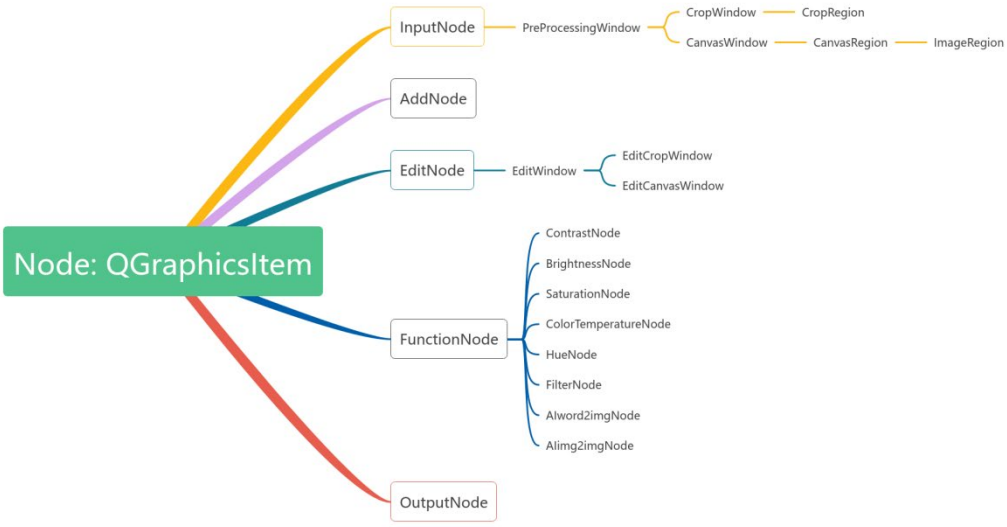

图 2-2. Node 类相关的派生和包含关系

2.2.1 节点、端口、连接线

节点由节点部件（NodeWidget）和输入输出端口（Terminal）组成。节点部件通过重写

基类的 execute()函数完成各节点的不同功能，对输入图像进行编辑并得到输出图像；输入输出端口与连接线（ConnectLine）相连，程序通过遍历连接线进行节点树的执行——上游节点将输出图像传递到输出端口，连接线读取输出端口的图像并传递到下游节点的输入端口，下游节点从输入端口读取图像。

2.2.2 输入节点、预处理窗口

输入节点能唤出预处理窗口（PreProcessingWindow）对输入图像做前期的处理。

预处理窗口中使用了堆叠窗口部件支持两个编辑模式，即裁剪（裁剪窗口，CropWindow）和移动、缩放、旋转（画布窗口，CanvasWindow）。二者均是 QWidget 的派生类。

裁剪窗口接受鼠标事件绘制裁剪框，裁剪确认后更新裁剪后的图像，同时更新 CanvasWindow 中显示的图像；提供 AI 抠图功能，调用 RemoveBg API 进行图片背景消除。

画布窗口接受鼠标事件调整图像在画布上的位置，通过滑块调整缩放比例和旋转角度。

确认修改之后，预处理窗口向输入节点传递编辑后的图像。此外，输入节点起到将输入图像尺寸统一为画布尺寸的功能。

2.2.3 基础调整节点

基础调整节点提供的编辑功能与输入节点相似，在图像更新逻辑上略有差异。在基础调整节点的裁剪过程中，不对原始图像做直接裁剪，而是记录裁剪框的位置和大小，在 execute()函数中计算裁剪后的图像。

2.2.4 调整节点

对比度、亮度、饱和度、色温、色调节点使用 OpenCV 中的相应函数进行图像相应属性的调整。

2.2.5 滤镜节点

相比其他调整节点，滤镜节点扩展了对 LUT 文件的加载和应用逻辑，能够对图像按照 LUT（Look Up Table）施加滤镜。具体实现方法为：

（1）通过 loadCubeFile()函数打开文件对话框，加载 .cube 格式的 LUT 文件（.cube 文件是 Adobe 公司规定的一种 LUT 文件格式，记录了 RGB 三维空间到另一 RGB 三维空间的映射，常用来作为图片滤镜或色彩空间转换）；

（2）通过 loadCube(const std::string &fileName) 函数读取文件内容并解析 .cube 文件中的 LUT 数据，将每行数据解析为 RGB 值，存储到 _lut 结构体中；

（3）通过 applyLUT(const LUT3D& lut, cv::Mat& image, float intensity) 函数根据指定强度应用 LUT 到图像；

（4）通过 applyLUTColor(const LUT3D &lut, const cv::Vec3f &color) 函数应用 LUT 到单

个颜色值。

2.2.6 人工智能抠图、文生图、图生图节点

为完成抠图、文生图、图生图功能，我们使用了三个 API，分别是：

功能	模型名	API 地址
移除背景	RemoveBg	https://api.remove.bg/v1.0/removebg
文生图	Dalle-3	https://api.ttapi.io/openai/v1/images/generations
图生图	MidJourney	https://ai.huashi6.com/aiapi/v1/mj/draw

我们使用 Qt Network 模块中的 QNetworkAccessManager 与 QNetworkRequest 进行网络通信。当按下 AI 节点中的“执行”按钮时，本地向服务器发送 API 请求。背景移除 API 传入 Base64 格式数据流，返回的是遮罩后的 Base64 格式图片，本地进行 base64 解析后转换 cv::Mat；文生图 API 传入文字 prompt，返回图片的 Url，由函数 downloadPhoto 下载到本地后读入到程序中；图生图 API 传入 Base64 格式数据流，返回的是四张图片的链接，经由函数 downloadPhoto 全部下载到本地后弹出图片选择界面，用户可从四张中选择一张。

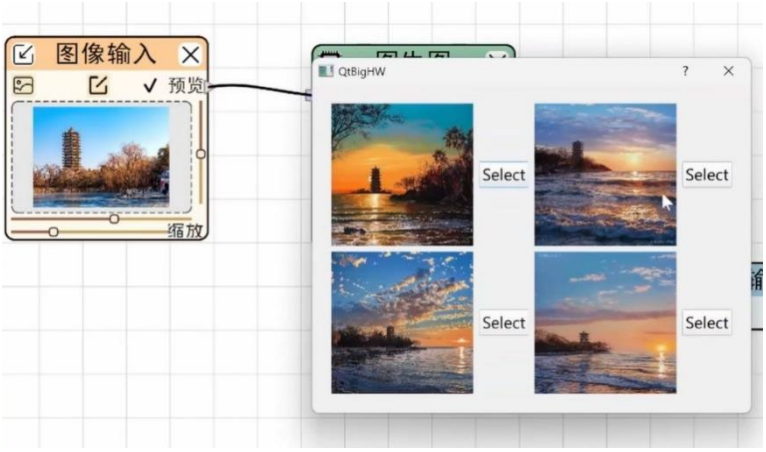

图 2-3. 以博雅塔为输入进行图生图后的样例展示

2.3 节点树的文件存取

2.3.1 文件的读入

调用 MainWindow 类的 openProject()函数从 ima 文件中读取信息，具体实现细节如下：

（1）打开文件对话框，让用户选择一个 .ima 文件。

（2）检查文件是否可读，否则显示错误信息。

（3）调用 newProject() 函数，重置当前项目。

（4）删除场景中的所有 outputNode 节点。

（5）使用 QDataStream 读取项目名称、画布大小、节点数量、节点位置信息、节点类

型和终端数量。

（6）读取所有连线的起始和终止位置，并在场景中重绘这些连线。

2.3.2 文件的保存

调用 MainWindow 类的 saveProject()函数提示用户选择保存路径和文件名，并调用 saveProjectToPath() 函数保存项目数据到指定路径的 ima 文件中，具体实现细节如下：

（1）项目信息保存：首先保存项目的名称、画布的宽度和高度。

（2）节点保存：遍历 _MainScene 中的所有节点，依次保存每个节点的位置、类型以及输入输出终端的数量。

（3）连线保存：遍历 _MainScene 中的所有图形项，筛选出类型为 ConnectLine 的对象，并新建一个 list 保存每条连线起始和终止终端的位置。

三. 小组成员分工

项目的代码部分分工如下：沈睿弘搭建了程序的整体框架，编写了主窗口和停靠部件，实现了 AI 节点和汇聚节点，以及输入节点的 AI 抠图功能，并实现了文件保存读取功能。陈睿哲编写了输入节点和基础调整节点，实现了一些图像数据类型转换的函数和编辑函数，并完成了所有部分的 UI 设计。高美珺编写了对比度节点、亮度节点、饱和度节点、色温节点、色调节点、滤镜节点。

小组报告、演示视频录制由三位同学共同完成，路演演讲由陈睿哲同学完成。

四. 总结与展望

4.1 项目回顾

Imaging 的灵感来自组员高美珺关于模拟场景下虚拟试衣的提议，加以市面上图像处理软件大多专业、复杂、难以日常使用的痛点，逐步完善成为一款图像处理软件的雏形，半学期里克服种种困难，终于成长为一段数千行的程序。

期间遇到的挑战数不胜数，诸如环境不匹配、程序结构复杂、对外部库的陌生、代码量巨大等等困难都对项目的推进造成了不小的阻碍。但截至目前，我们已经完成了最初功能设计中基本所有的设想，顺利实现了一个集简洁、美观、便捷、智能于一身的节点式图像处理软件。

本次项目是我们第一次使用 Qt 编写较大的程序，也是我们第一次接触 OpenCV 库，项目中的文件处理、图像编辑、事件循环等等也都是新的知识点；此外，这也是我们第一次自行完成规模如此大的项目，使用 Github 进行多人协作，协调各自编写的部分，在分工后整合各项功能，都是前所未有的体验。总而言之，程设 Qt 大作业的实践中，我们收获颇丰。

4.2 项目优势

4.2.1 图像处理功能丰富

Imaging 软件不仅支持最基础的图像处理功能，还支持基于 AI 的高级功能。这使得用户可以在一个平台上完成从基础调整到高级处理的全流程操作，极大地提升了用户体验和工作效率。

4.2.2 节点式工作流程

相比于市面上常见的图像处理软件，例如 Photoshop 等，Imaging 最显著的特点是图像处理的模式。举例而言，对于传统的层级式的操作界面，如果在多项操作后忽然发现需要更改之前操作的某些参数，如一级调色的某些参数、色彩配置文件的映射等时，往往需要一直撤销后重新制作，不利于美术人员的实时修改。而节点式操作界面保留了所有计算过程与计算参数，可以根据需要任意调节，解决了这一问题。

在节点式图像处理的操作界面中，每一步编辑都以节点的形式显示，每一步的运行结果都在节点上呈现，更便于及时的修改，也使图像处理的流程一目了然。

4.3 展望

Imaging 是目前由小组成员们自行完成的规模最大的项目。从前期灵感收集、形成方案，到着手搭建环境、实现代码，再到最后测试改进、整理报告，对我们都是前所未有的挑战。虽然我们已经完成了绝大部分目标，但因时间紧、任务重，加以经验不足，仍有许多值得改进的地方。

就功能而言，输入节点提供的抠图功能仅限于 AI 抠图和矩形框截取，没有更加自由的手动抠图功能；一些调整节点的滑块数值范围超出了正常图像编辑的范围，造成图像处理的困难；主窗口上的撤销功能并不完善，目前不支持对删除节点、删除连接线等操作的回退。就 UI 友好程度而言，目前为了方便编辑和确保视效稳定，Imaging 的窗口大小仍然是固定的，其中的各个部分也大多不支持改变大小，使用体验较为呆板。

在 Imaging 发布之后，我们还会进一步处理其中的 bug 和有待改进的部分。在未来的学习实践中，我们也会继续积累经验，进一步完善自己编写项目的思维和能力。