多模态医学影像配准、分割与可视化系统

软件架构文档

版本 **<1.2>**

修订历史记录

日期	版本	说明	作者
<日/月/年>	<x.x>	<详细信息>	<姓名>
2020.10.23	1.0	初版架构文档	张诏崴
2020.10.25	1.1	添加核心算法	赵语云
2020.10.25	1.2	添加前端技术框架	罗媚 李陈豪

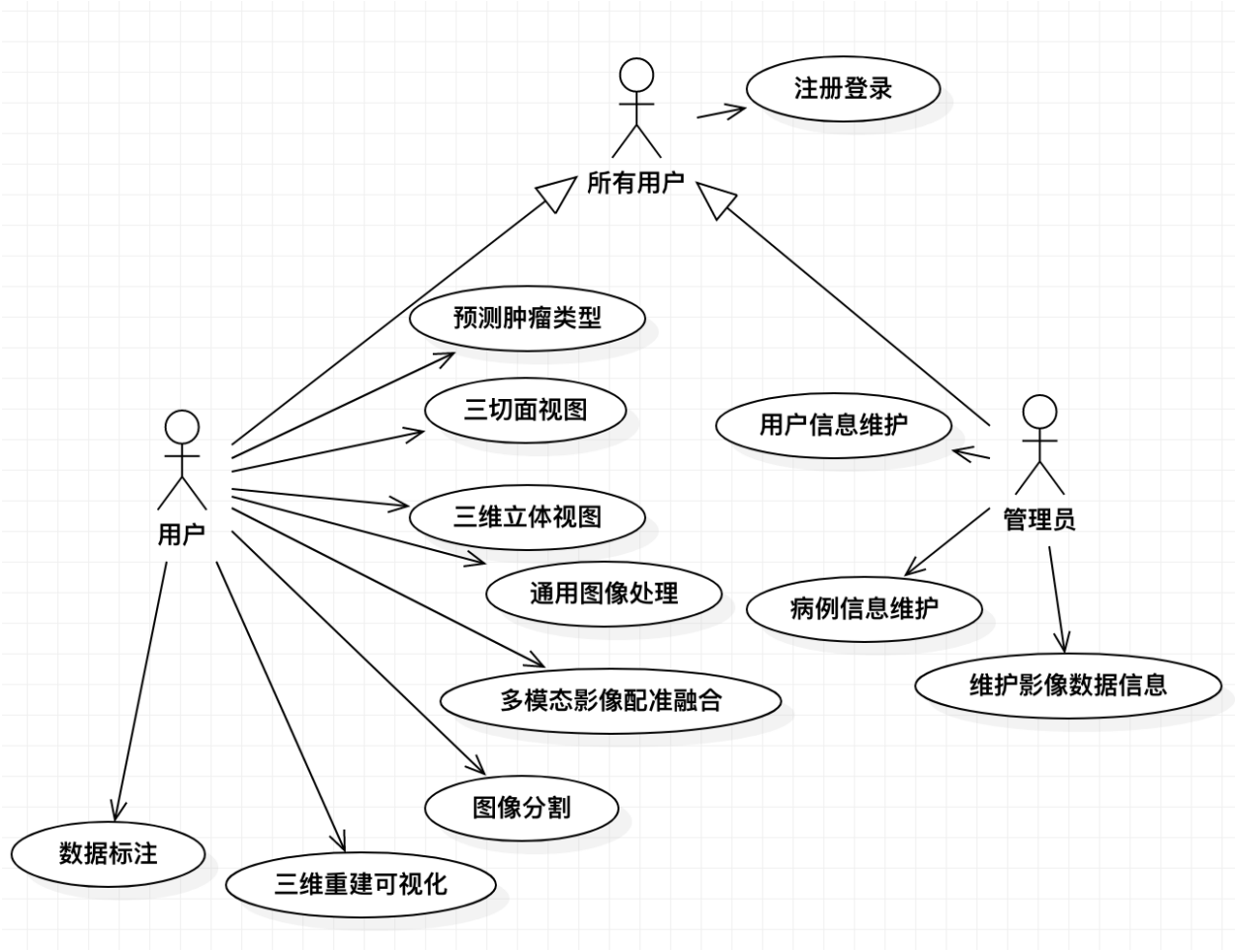
目录

1. 简介

1.1. 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

2. 用例视图



本系统的用户主要包括数据维护的管理员，以及客户端的使用用户。所有用户都需注册登录本系统，注册信息储存于系统的服务器后端，并以此区分用户在本系统中的角色定位。

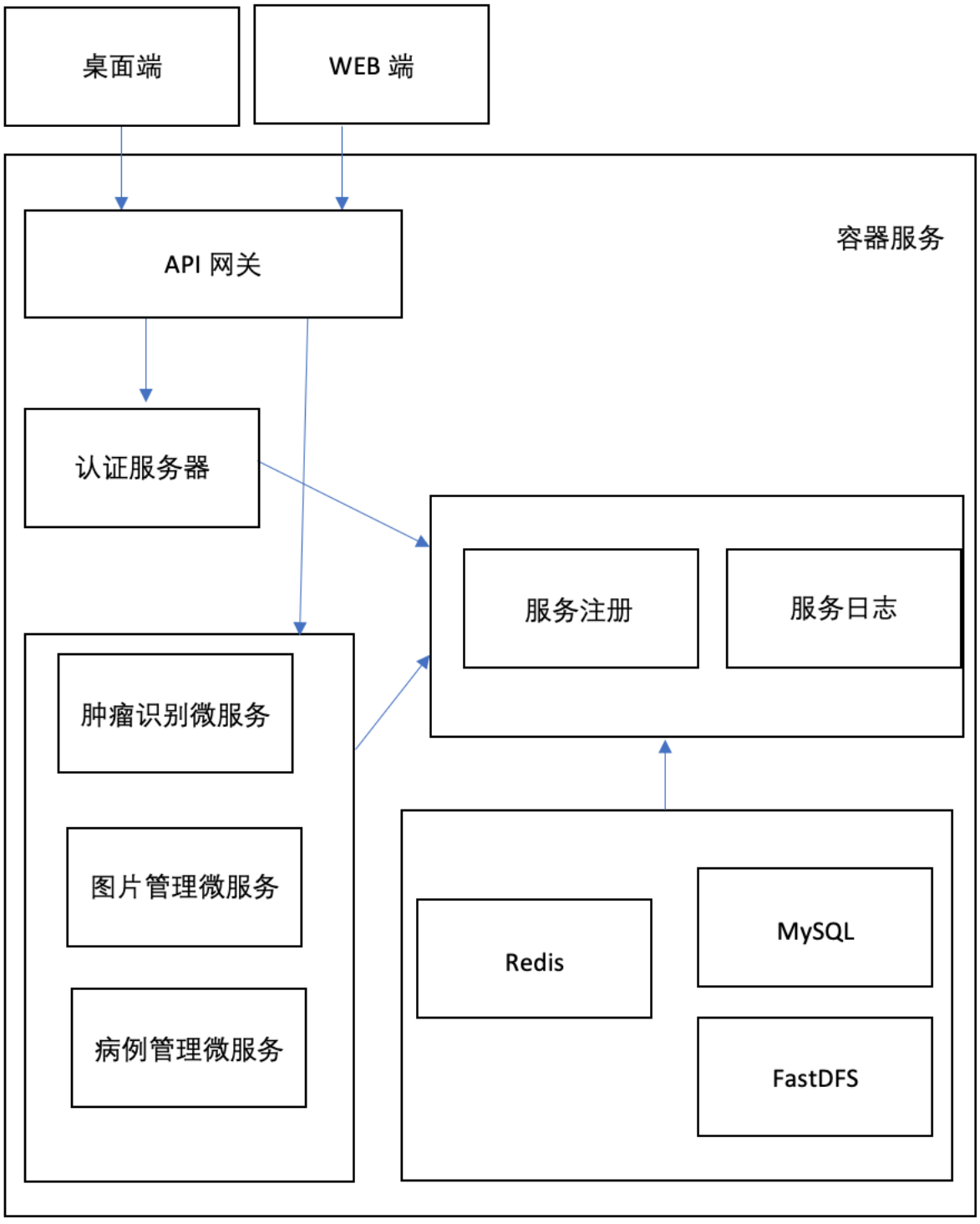
客户端用户主要使用医学影像处理相关的功能，其中包括客户端本身和需要服务器提供支持的两部分功能。客户端本身提供基础图像处理相关的算法功能，包括对医学影像的三切面视图，对病例的三维立体视图，对不同模态的医学影像的配准融合功能，通用的图像处理包括平滑、边缘检测等，对病灶的三维模型重建可视化，对图像内容的分割。需后端支持的功能包括，针对病例影像进行肿瘤类型的预测，需服务器提供机器学习算法模型；针对新病例进行数据标注，例如肿瘤位置、类型等，并将结果存储于服务器。

管理员用户在本系统中需对用户信息进行维护，对病例数据进行管理，对病例的影像数据进行管理。

3. 逻辑视图

3.1. 概述

系统采用基于容器的微服务架构。用户通过桌面客户端访问系统，管理员通过 WEB 端浏览器访问系统。所有的服务端组件和客户端都在 docker 容器中运行。所有请求通过 Nginx API 网关，网关判定需要进行认证的请求通过认证服务器获得令牌，之后才能访问肿瘤识别服务，图片管理服务以及病例管理服务等具体服务。所有的服务都要在注册中心进行注册，在日志系统中监控服务的日志。



3.2. 第三方库

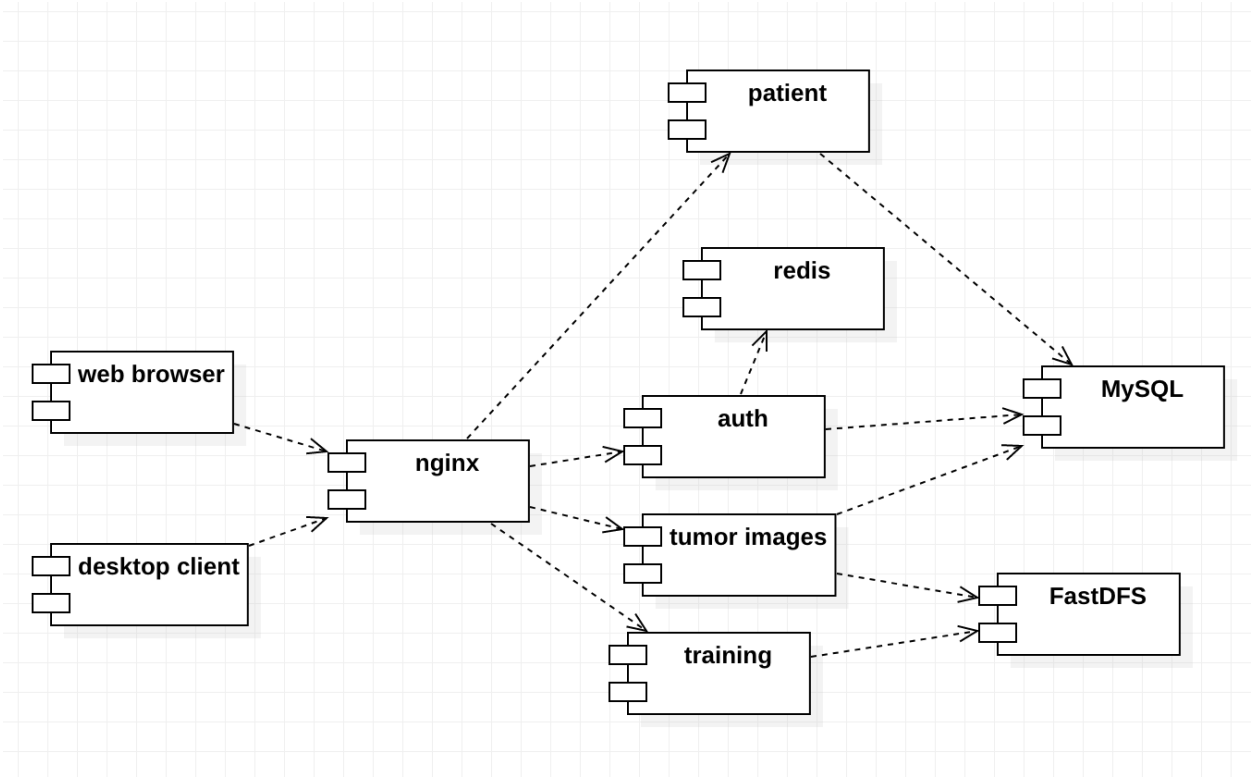
在客户端开发中，需要对用户界面进行设计开发，需要完成对医学影像处理相关的算法开发，以及完成与远端后台的交互接口。客户端的界面开发主要利用 QT 的 core gui 模块，包括 widget 库等进行开

发。医学影像相关核心算法需要用到 VTK、ITK 等库辅助开发。后端通信上利用 QT 的 network 模块进行开发，使用 QNetworkAccessManager 等库进行 http 通信的实现。

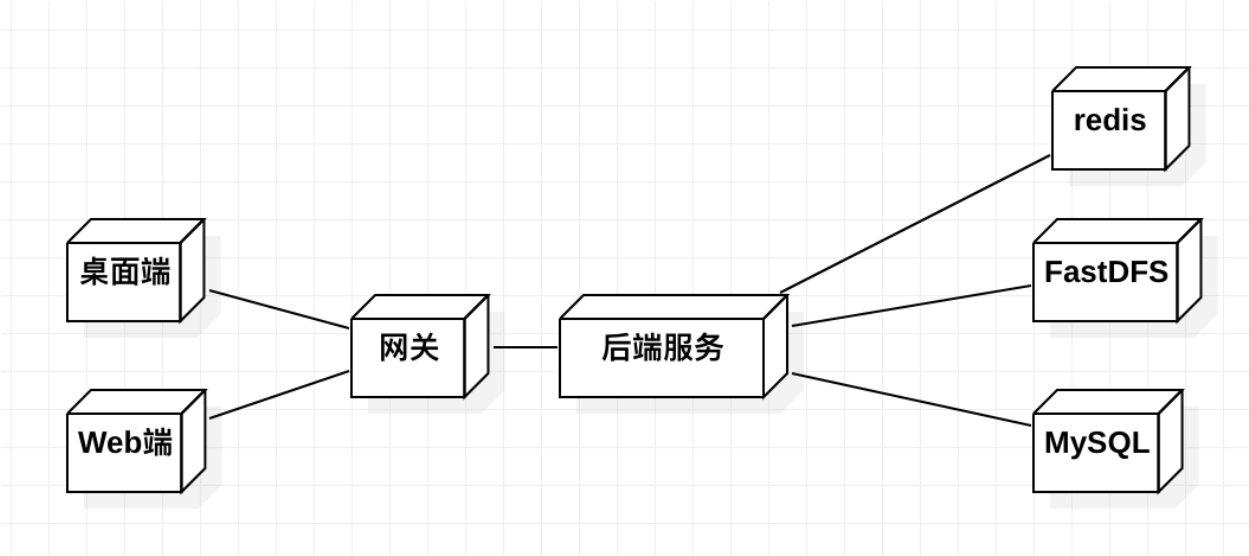
4. 进程视图

服务端包含 nginx，病例管理服务，认证服务，肿瘤识别服务，图片服务，MySQL 数据库，Redis，EFL，consul 注册中心等进程。所有进程都是在 docker 中运行。

客户端包括用户注册服务，图像可视化，图像配准处理服务，以及与服务端通信，数据传输等服务。医学图像处理使用 ITK 和 VTK 库进行算法开发，服务端通信使用 QT 的 network 模块开发，主要使用 QNetworkAccessManager 库进行接口实现，UI 窗口设计主要通过 QT 的 core gui 模块开发。

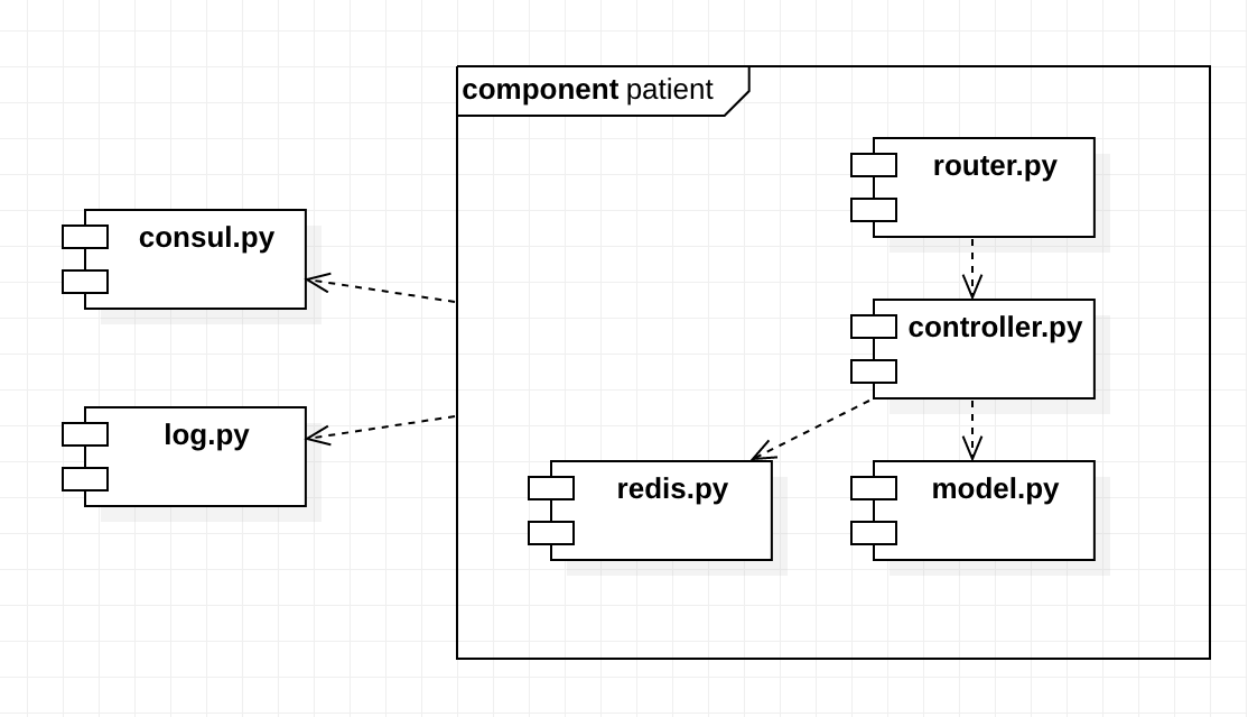


5. 部署视图



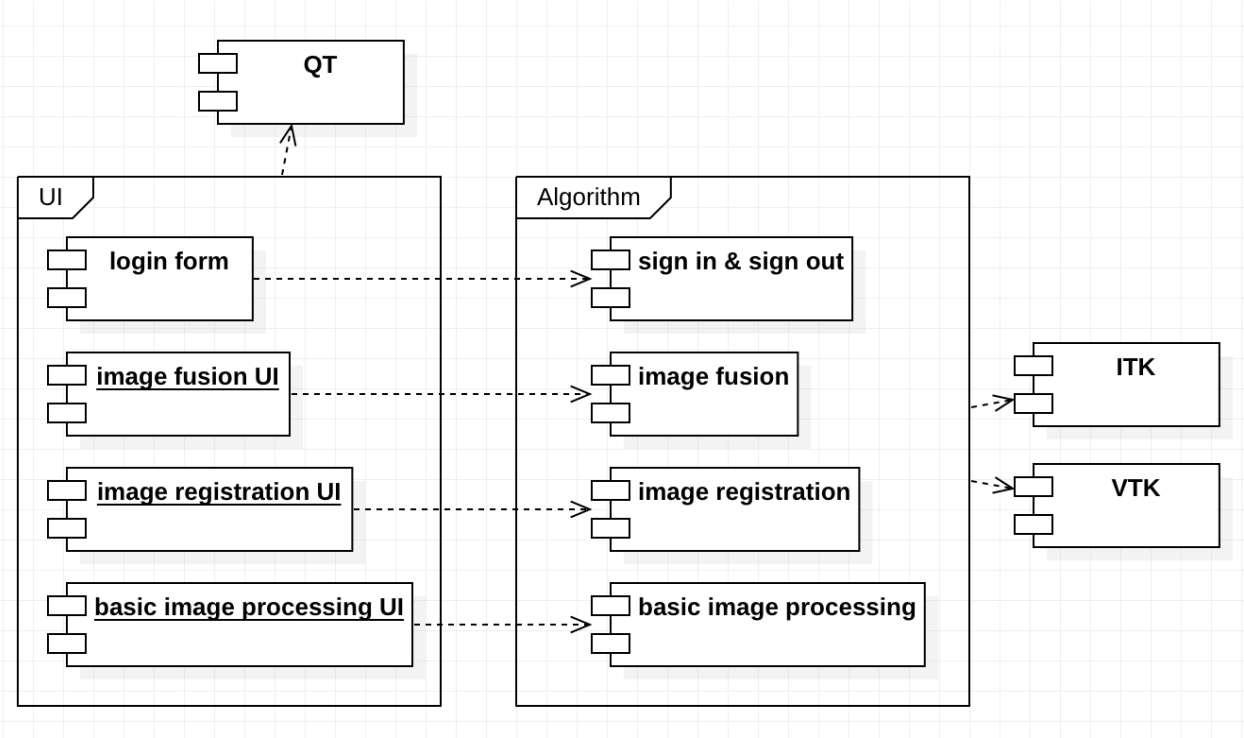
6. 实现视图

服务端的具体服务采用 MVC 方式设计，服务包含 router.py（用来处理路由），controller.py（处理具体逻辑）model.py（处理数据实体）redis.py（处理 redis 缓存）。每个服务都会和外部的 consul.py（注册中心）和 log.py（日志中心）对接，之后通过网关暴露给客户端。



客户端为基于 QT 框架的 PC 端应用。客户端主要分为 UI 部分与算法部分，两部分相对耦合。UI 部

分负责 PC 端的界面展示并提供对应的前端接口。算法部分主要负责基于本机的医学图像处理以及与系统远端后台的信息交互。基础的图像处理等功能将在客户端实现，具体通过调用处理医学图像的 ITK 与 VTK 库实现。与系统后端交互部分，主要通过 QT 提供的 network 框架进行处理，主要使用 QNetworkAccessManager 库进行实现接口，与服务端通信采用 http 通信。

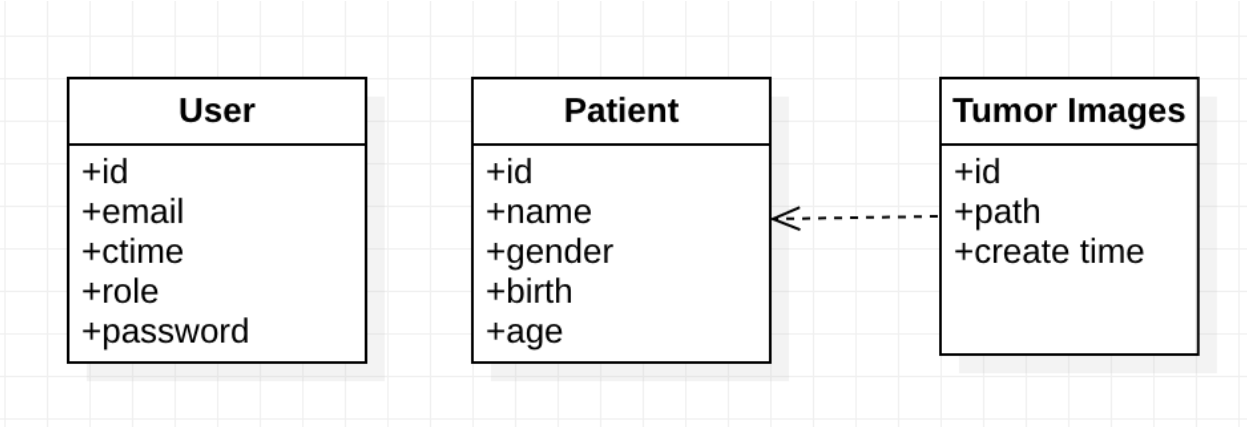


7. 技术视图

系统采用支持 Windows 和 MacOS 作为操作系统，关系型数据库使用 MySQL，文件服务器采用 FastDFS，深度学习框架采用 PyTorch，服务端开发语言采用 Python，框架使用 Flask，桌面端使用 C++ QT 进行开发，Web 端使用 React 框架开发。


桌面端
C++ Qt
Web 端
React
服务端框架
Flask Python
深度学习框架
PyTorch
关系型数据库
MySQL
文件服务器
FastDFS
操作系统
Windows，MacOS

8. 数据视图



9. 核心算法设计

9.1. 多模态影像配准算法设计

（1）确定固定图像 f 与浮动图像 m，其中浮动图像为待配准图像。根据配准任务种类选择合适的配准测度函数 S、插值器、几何变换 T 以及优化器类型。初始化几何变换 T。

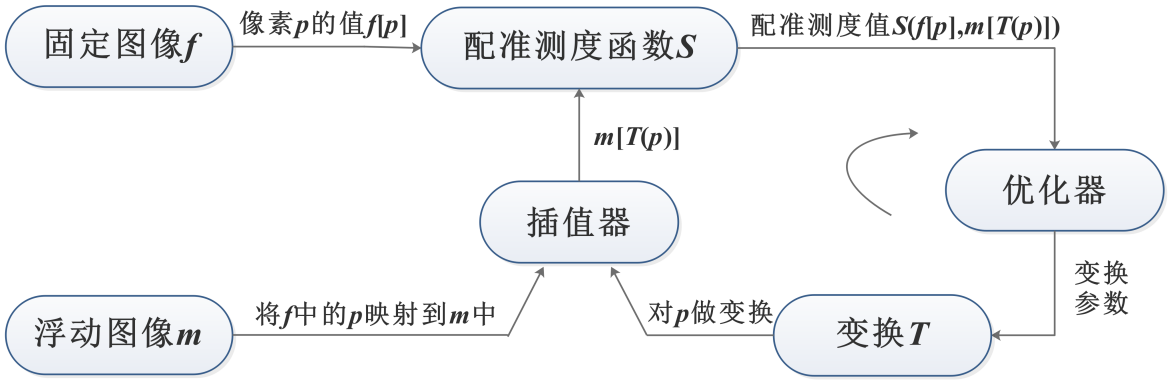
（2）对固定图像 f 中每个像素点 p 做几何变换 T，映射到浮动图像 m 中的 q 点，并通过插值器得到浮动图像 m 中 q 点处的灰度值 m[T(p)]；

（3）计算固定图像 f 中 p 点的灰度值 f[p]与 m[T(p)]的配准测度值 S(f[p], m[T(p)])；

（4）使用优化器对配准测度值进行优化，得到优化后的几何变换参数，更新几何变换 T 的参数；

（5）重复执行步骤 2-4，直到配准测度值达到最优值，停止迭代。

（6）使用优化结束后的几何变换 T 对图像进行变换得到配准后的图像。

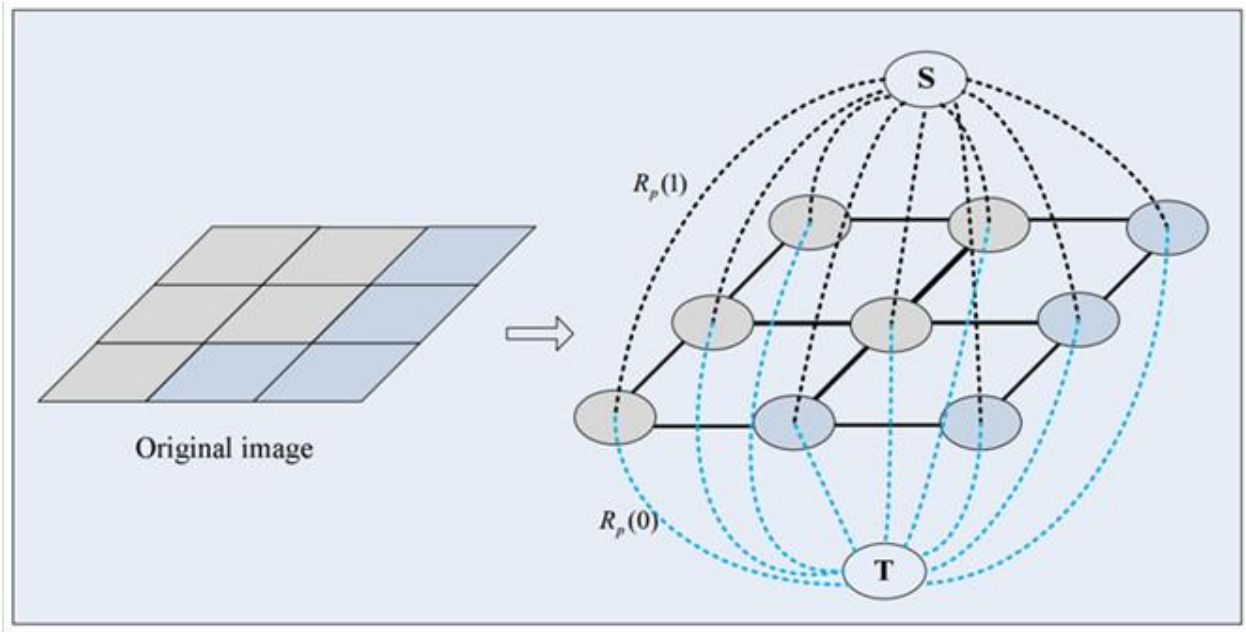


9.2. GraphCut3D 算法设计

Graph Cut 算法在图论中普通图的基础上增加了两个顶点与第二类边，称这两种顶点为终端顶点，分别表示为“S”与“T”。图中的其他顶点都会与这两个顶点相连接，连接成的边即为第二类边。总结 Graph Cut 算法中顶点与边的类型如下：

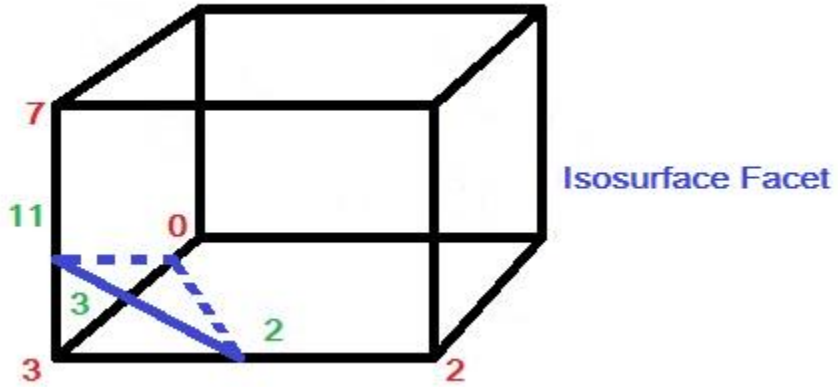
（1）第一类顶点与边：第一类顶点对应图像中的所有像素点。图像中每两个邻域像素点（即邻域顶点）之间都会有一条边连接二者，这种边为第一类边，称其为 n-links。

（2）第二类顶点与边：第二类顶点为两个终端顶点“S”与“T”，分别代表了前景与背景。所有的第一类顶点都会与第二类顶点相连接，代表这些顶点所属的类型是前景还是背景，其连接而成的边即为第二类边，称其为 t-links。



9.3. 面绘制 MarchingCubes 算法设计

具体实现思路为：首先将三维数据场中的体素作为基本单位，对体素的 8 个顶点做编号 1-8；每个顶点都有其标量值（在 CT 图像中为每个像素点的 HU 值），若顶点上的标量值大于用户设置的需要提取的等值面的值，则表明该顶点在等值面之外，将其标记为“0”，反之，若某顶点的标量值小于等面值的值，则将其标记为“1”；每个体素都有 8 个顶点，因此共存在有 28 种情形，图 4-5 展示了其中的 15 种基础情形，其他情形则都可通过对这 15 种进行旋转、翻转等操作实现。



10. 质量属性的设计

1） 易用性

提供操作指南文档，普通用户能够快速上手使用。软件提供帮助选项，用户能够在软件操作界面查看操作指南，并且用户能够在 2 小时内熟悉对软件的使用。

2） 可靠性

用户可用时间百分比为 99.99%。 MTBF：平均故障间隔时间为 120 天。MTTR：平均修复时间为 10 小时。系统每周进行一次备份。

影像处理的算法精度达到实用的水准，影响配准的精度误差控制在物理空间的 2 毫米以内。

3） 性能

在用户平均网络硬件水平的情况下，每个操作相应时间不超过 2 秒。

任何问题造成未响应，系统在 5 秒后在页面上显示未响应等提示信息。

吞吐量：系统能同时处理 50 事务/秒。

服务器内存为 4GB， CPU 2 核， 磁盘 50GB， 公共带宽 100 MB/s。

4）可支持性

编码标准和命名约定参照 Google 的开发规范。系统有统一的日志输出机制和报告机制。

5）可扩展性

采用微服务的开发架构，需要扩展的时候能够添加新的服务并在注册中心上进行注册即可，不需要影像现有的软件架构和内容。