管道机器人裂纹检测系统设计说明

浙江大学 2020 年科研训练研究报告（B）

项目编号: X20200352

项目名称: 能适应弯道和垂直爬行的管道机器人研究

项目负责人: 张文博 (3180103098) 1792330061@qq.com

项目成员: 王一帆 (3180104461)

卢成宇 (3180104328)

指导老师: 王宣银

目录

1. 开发背景

管道机器人需要能够实现管道内的基本探测任务，如裂纹检测、障碍清理等，可以使用机器视觉的方案，通过传统图像处理算法或深度学习的神经网络对摄像头拍摄到的图像进行处理。管道探测任务需要实时处理摄像头图像，而机器人上的嵌入式系统通常对图像的处理能力有限，很难保证图像处理的实时性。因此，实现管道内部探测任务，需要开发一套软件系统，能够将摄像头拍摄图像实时发送到计算机，并在计算机上显示结果。

2. 需求分析

实现图像实时传输分析，主要需要完成两个方面的设计，一是图像实时传输的实现，二是图像实时处理的实现，现分别对这两个方面的需求进行分析。

2.1 图像实时传输需求分析

1）图像质量：对于管道内的裂纹检测，图像分辨率与清晰度是能否识别出细小裂纹的关键，因此图像质量是软件系统设计需要考虑的关键问题，图像的分辨率不应小于 720P。

2）传输延时：对于裂纹检测系统，对图像传输的实时性要求不高，可以接收图像传输存在一定的延时，但也应当设法减小图像传输延时，传输延时不应超过 10s。

3）图像传输：图像发送设备为机器人上的嵌入式系统，需要针对嵌入式系统开发视频发送软件；图像接收设备为计算机，也需要开发视频接收的相关软件；同时需要规定视频发送接收的协议。

4）传输方式：管道机器人工作在管道内部，通常需要使用无线传输视频的方案。

5）多设备：管道探测任务通常会多个机器人同时进行，因此软件系统需要实现多个机器人同时探测时的图像传输分析。

2.2 图像实时处理需求分析

1）处理结果：需要得到图像中是否有裂纹以及标记出裂纹位置。

2）实时处理：需要实时处理图像数据，单帧图像处理时间应在 1s 以内，因此对于计算机对图像处理性能有一定要求。

3. 系统设计

3.1 总体设计

考虑多设备情况，图像传输应采用服务器-客户端分离的方案，即通过流媒体服务器，多个机器人可以同时向服务器推流，检测系统从服务器拉取对应机器人的视频流，实现多个机器人同时探测时的图像实时传输与分析。考虑到图像处理对计算机性能的要求，图像处理的理想方案也是将服务器-客户端分离，图像处理模型可以部署在一台高性能的远程服务器上，本地设备隐形的检测系统负责将拉取的视频流发送到模型服务器，模型服务器处理图像并将结果返回，检测系统接收返回结果并进行显示。

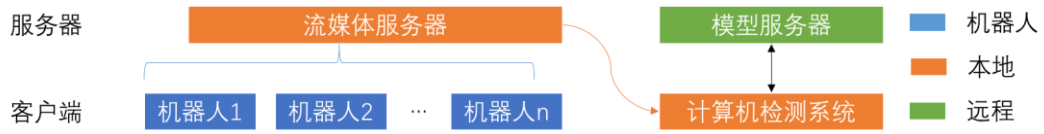

图 1 系统总体方案设计

裂纹检测系统的总体设计方案如图 1 所示。其中流媒体服务器对计算机性能要求不高，可以运行在本地，而模型服务器对计算机性能要求较高，部署在远程。

3.2 视频传输方案设计

目前主流的实时视频流传输协议主要有 HttpFlv，RTMP，HLS 等，各协议对比如表 1 所示。

表 1 实时视频流传输协议对比

协议	HttpFlv	RTMP	HLS
传输方式	Http 长连接	TCP 长连接	Http 短连接
视频封装格式	FLV	FLV	TS
延时	短(1-3s)	短(1-3s)	长(20s)
播放支持	支持 H5 播放	需要 Flash，不支持 H5	支持 H5 播放

对于计算机检测系统软件，理想情况为拉取支持 H5 播放的视频流，使用现有的 H5 播放器（如 Flv.js）框架，可以大大减少软件开发的成本；而对于机器人设备，如果需要实现对 Http 协议传输的支持则会加大软件的开发成本，理想的情况是使用基于 TCP 传输的协议。为了解决这一矛盾，降低软件开发成本，可以考虑使用流媒体服务器进行转流，即机器人设备使用 RTMP 协议向流媒体服务器推流，由流媒体服务器将 RTMP 协议视频流转换为 HttpFlv 协议视频流，计算机检测系统拉取 HttpFlv 协议视频流。

机器人 RTMP 协议推流可以基于 yasea 开源库实现，根据检测系统软件运行环境，流媒体服务器选择支持 NodeJS 的服务器框架，如 node-media-sever，检测系统通过 flv.js 等 H5 播放器框架实现实时视频的播放。整个视频传输方案设计如图 2 所示。

图 2 视频传输方案设计

3.3 机器人图像采集设计

现提出一种使用智能手机作为机器人图像采集设备的方案。目前主流的智能手机都配有高清摄像头，且可以使用网络、蓝牙、串口等多种方式进行通信，因此将智能手机安装在机器人上并使用智能手机作为机器人的图像采集设备在理论上是可行的。现以 Android 系统智能手机为例设计图像采集发送方案。

Android 系统可以通过调用 Camera 相关的 API 获取摄像头，使用 yasea 库将摄像头拍摄图像转换为 RTMP 协议视频流并发送到流媒体服务器。整个图像采集系统设计如图 3 所示。

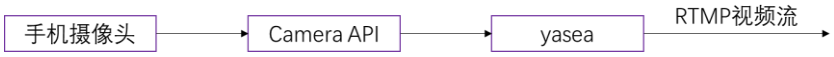

图 3 机器人图像采集方案设计

3.4 检测系统软件设计

考虑到软件开发成本等因素，现基于 Electron 框架开发检测系统软件。Electron 框架实现使用 html+javascript+css 开发桌面应用，相比于其他开发方式（如 Qt，Java 等）其生态环境更好，可用框架更多，能够缩短软件开发周期。其不足是程序运行在 Chrome V8 引擎上，因此程序运行效率相比于使用 C 语言开发的程序低，且程序整体体积较大。

在检测系统总体方案设计中，提出了流媒体服务器部署在本地的方案。如果流媒体服务器部署在本地则可以将流媒体服务器整合到检测系统中。可以使用 flv.js 等播放器框架实现实时视频的播放，使用 html5 的 canvas 相关 API 可以截取视频流中的单帧图像，向模型服务器发送 ajax 请求实现与模型服务器的通信，canvas 相关 API 同时可以将服务器返回结果处理为图像并显示。检测系统软件设计如图 4 所示。

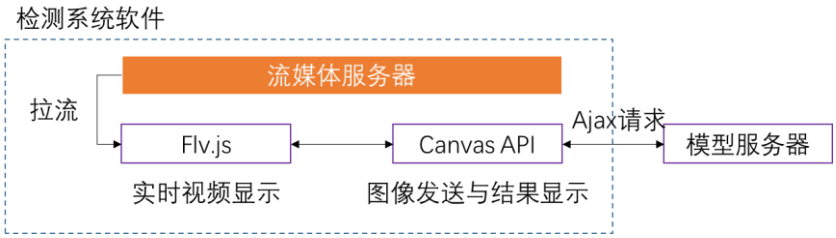

图 4 检测软件设计

3.5 模型服务器方案设计

使用传统的图像处理方式进行图像裂纹检测通常不能达到令人满意的准确度，因此需要使用深度学习的神经网络对图像进行处理，即需要在远程服务器部署网络模型，处理来自检测系统发送的图像请求。

现主流的模型服务器框架有 Tensorflow Serving，TorchServe 等。使用 Tensorflow 训练的模型通常使用 Tensorflow Serving 进行部署，而 Pytorch 训练的模型通常使用 TorchServe 进行部署，以避免模型转换可能带来的错误。现使用 TorchServe 对模型进行部署。

TorchServe 提供了模型预测接口与服务管理接口。通过模型预测接口可以向服务器发送待处理的图像，服务器处理图像并返回预测结果；服务管理接口可以查看当前服务器的状态、运行的模型，添加或移除现有模型等操作。模型服务器的方案设计如图 5 所示。

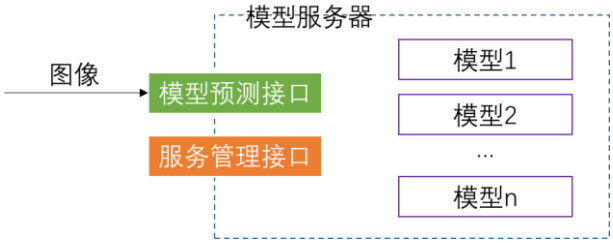

图 5 模型服务器方案设计

3.6 方案设计总结

整个裂纹检测系统设计方案如图 6 所示。

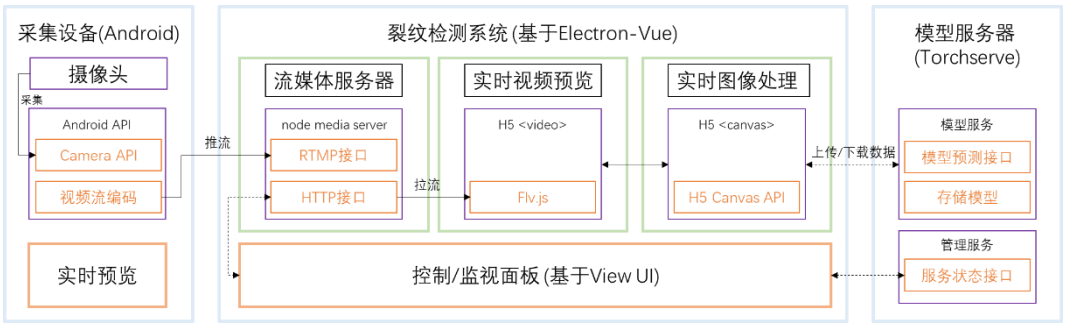

图 6 裂纹检测系统方案设计

4. 软件开发

4.1 Android 图像采集软件开发

Android 程序使用 Android Studio 进行开发，使用 Android 相关 API 配合 yasea 库可以实现实时视频的拍摄与上传，具体可见项目地址 https://github.com/WangYf-zju/CrackDetectionMobile。

4.2 裂纹检测系统桌面软件开发

裂纹检测系统基于 Electron 框架进行开发，需要首先安装 nodejs。为了使编写软件 UI 界面更加灵活高效，需要在项目中引入 Vue 框架并配合 ViewUI 框架。具体可见项目地址 https://github.com/WangYf-zju/CrackDetectionSystem。

4.3 模型服务器搭建

TorchServe 可在 Windows 或 Linux 等操作系统上运行。若直接安装 TorchServe 需要配置运行环境，也可以在 docker 容器中运行，免去环境配置相关操作。使用 docker 进行模型服务器搭建具体步骤可见项目地址 https://github.com/WangYf-zju/CrackDetectionSystem/blob/master/docs/model_server_quick_start.md。

5. 软件测试

5.1 Android 图像采集软件测试

Android 程序主界面如图 7 所示。输入流媒体服务器地址，并设定流名称，点击连接即可开启手机摄像头；若连接上流媒体服务器则开始向服务器推流。

图 7 图像采集软件

5.2 模型服务器测试

启动 docker 容器并在容器中启动 TorchServe，使用 curl 工具向服务器模型预测接口发送一张图像进行测试，并将结果保存到文件 res 中。可以观察到服务器返回了一个 224×224×2 的数组，保存了每个像素点为各个类别的概率。

服务器中运行的模型是一个 FCN 全连接神经网络，因此可以向服务器发送任意大小的图像进行测试。

5.3 裂纹检测系统桌面软件测试

裂纹检测系统桌面软件主界面如图 8 所示。主界面分为 4 个区块，分别为视频/图像结果预览区，流媒体服务器管理面板，视频流管理面板，模型服务器管理面板。流媒体服务器管理面板可以控制流媒体服务器的开关，并监视服务器当前状态；视频流管理面板可以启动/停止视频流的实时预览，切换视频流，监视视频流信息；模型服务器管理面板可以设置服务器地址，选择预测模型，监视服务器状态。

图 8 裂纹检测系统桌面程序

5.4 系统整体功能测试

现以使用本地局域网为例测试整个检测系统的功能。系统中各设备的网络地址如表 2 所示。

表 2 设备网络地址表

设备名称	设备类型	设备地址
Router	无线路由器	192.168.1.1
Model Server	模型服务器	192.168.1.101
my-PC	本地计算机设备（运行软件系统与流媒体服务器）	192.168.1.102
Phone1	Android 虚拟手机	127.0.0.1
Phone2	Android 真机 1	192.168.1.103
Phone3	Android 真机 2	192.168.1.104

图 9 实时图像传输测试

在流媒体服务器控制面板点击启动按钮，流媒体服务器启动并且显示服务器状态。在虚拟手机与真机上启动 Android 图像采集软件，连接流媒体服务器，服务器地址填写 192.168.1.102:1935（RTMP 服务运行在 1935 端口上），应用名称分别填写/app/main，/app/test1，/app/test2，在视频流控制面板可以观察到设备接入，点击启动按钮开始预览手机摄像头图像，点击切换设备按钮可以切换预览设备，如图 9 所示。

在模型服务器控制面板点击服务设置按钮，填写服务器地址，若连接上服务器程序自动列举出服务器上的模型，选择模型，点击启动按钮可以得到图像实时分析结果的预览，如所示。

图 10 模型服务器设置

6. 设计总结

本次项目完成了裂纹检测系统的设计，包括 Android 手机图像采集程序，裂纹检测系统桌面应用程序以及模型服务器构建程序。程序设计中主要的创新点有：

1）使用 Electron 框架开发桌面应用程序，缩短了软件开发周期；

2）采用服务器-客户端分离的模式设计系统，相关服务部署迁移更加灵活，且支持多设备通信，而不仅仅支持点对点的通信。

程序中值得改进之处有：

1）系统桌面软件通过 h5 播放器框架先解析视频流数据，然后将得到的图像发送到模型服务器进行处理，会产生较大的延时，理想的方案是流媒体服务器直接将视频流转换为图像并发送到模型服务器，但这种方案没有现成的框架，需要自行编写底层转流相关程序，将会大大增加软件的开发成本。

2）模型服务器现部署的模型是对单帧图像进行处理的模型，对于视频流数据，训练一个能够处理图像序列的模型是更优的方案。

3）流媒体服务器暂未完全整合到检测系统桌面程序中，因此暂时要求用户预先安装好 NodeJS。

这些问题将在未来一段时间内进行优化。