浙江大学

**科研和工程中的C++编程开发报告**

**小组成员：高涛 李逸婷 毛晨炀 徐超颖 徐可添**

目录

[摘要 2](#_Toc394087954)

[课题设计 2](#_Toc394087955)

[开发环境 2](#_Toc394087956)

[开发分工 2](#_Toc394087957)

[编译运行 2](#_Toc394087958)

[详细设计 3](#_Toc394087959)

[程序结构 3](#_Toc394087960)

[主要实现 5](#_Toc394087961)

[开发日志 11](#_Toc394087962)

[程序运行 12](#_Toc394087963)

[程序使用说明 12](#_Toc394087964)

[程序运行示例 13](#_Toc394087965)

[附件（GitHub地址） 18](#_Toc394087966)

# 摘要

## 课题设计

目的：主要用于为后续使用Deep Learning训练交通标志检测器提供更多的训练样本；也可用于其他视频中物体（如人脸）的追踪。  
输入：一段含有交通标志的视频，交通标志出现时所在的帧与位置信息  
输出：该交通标识所出现的所有帧和位置信息、该交通标志在这些帧的截图

## 开发环境

操作系统：Ubuntu Desktop 14.04  
集成开发环境：Qt Creator 3.0.1 (GCC 4.8.2)  
依赖库：Qt 4, Boost 1.54, OpenCV 2.4, Eigen 2  
工程管理：CMake 2.8  
版本控制：GitHub

## 开发分工

* 高涛：图形界面，应用程序框架
* 李逸婷：图像读写与处理
* 毛晨炀：结果标记与保存、测试
* 徐超颖：视频读取、测试
* 徐可添（组长）：应用程序框架、追踪算法（基于Struck, ICCV 2011）

## 编译运行

1. 复制项目到本地：git clone

<https://github.com/Zhejiang-University-GKC/traffic-sign.git>

1. cd traffic-sign
2. 可以在外部编译，先新建一个文件夹：mkdir build
3. 拷贝配置文件：cp config/\* build/
4. cd build
5. 如果需要保存追踪结果的截图，需要手动建立image文件夹：mkdir image
6. 编译：cmake .. && make
7. 运行：./main
8. 点击 Video 选择视频文件
9. 点击 Run 选择需要追踪物体的初始信息并运行

*说明：由于依赖库的安装目录不同，可能需要在编译前修改 cmakelists.txt*

# 详细设计

## 程序结构

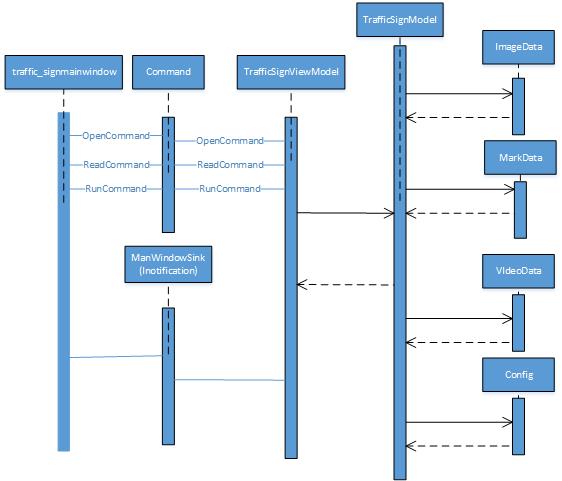


图 1 类关系图

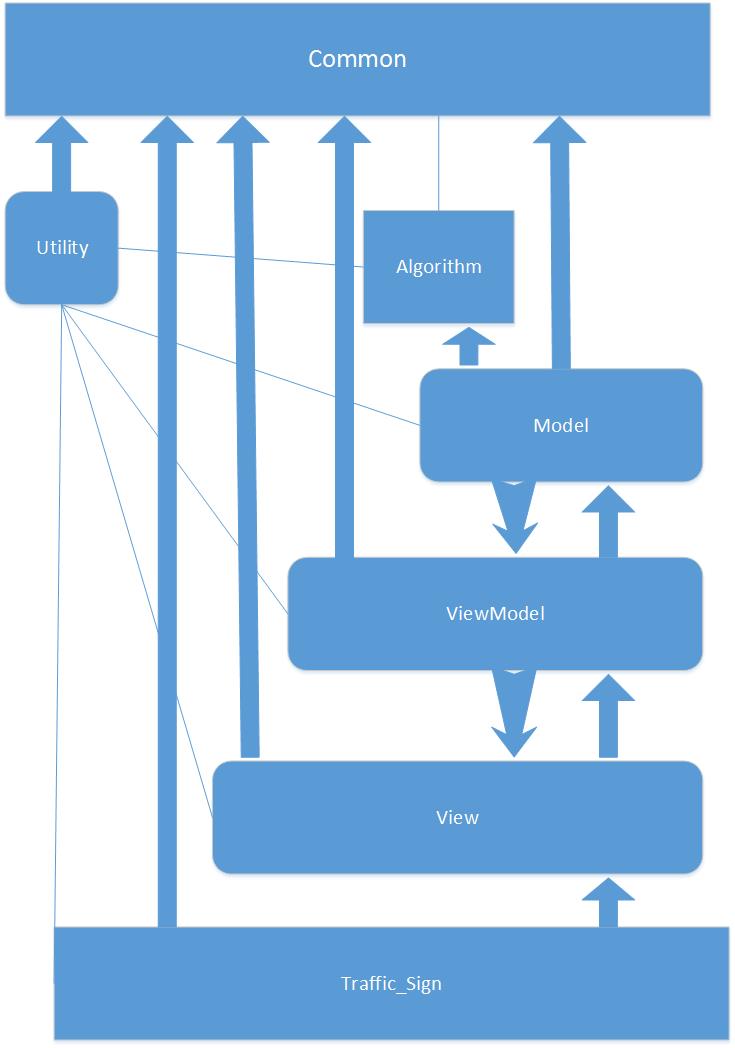


图 2 层次结构图

## 主要实现

###### Algorithm

**1. Tracker.h**

**1.1 Tracker类**

执行根据所给配置文件分析跟踪目标标标志，记录该标志所在的长方形区域功能。

成员变量：

Tracker::const Config& m\_config

保存配置文件信息的变量m\_config

Tracker::bool m\_initialised

记录是否被初始化过的标志变量m\_initialised，1表示已初始化，0表示未初始化过

Tracker::std::vector<Features\*> m\_features

Vector中保存了特征信息Feature，容器名是m\_feature

Tracker::std::vector<Kernel\*> m\_kernels;

算法中所使用的核的记录容器m\_kernels

Tracker::LaRank\* m\_pLearner;

采样信息，以供给下次分析提供样本信息，让机器学习

Tracker::FloatRect m\_bb;

记录跟踪分析出的目标标志所在的矩形区域信息

成员函数：

Tracker::Tracker(const Config& conf);

以配置文件作为构造函数的参数，并进行构造初始化

Tracker::~Tracker();

析构函数

void Tracker::Initialise(const cv::Mat& frame, FloatRect bb);

初始化每一帧中标志所在矩形区域信息

void Tracker::Reset();

复位Tracker，清空信息

void Tracker::Track(const cv::Mat& frame);

跟踪每一帧中的标志

inline const FloatRect& Tracker::GetBB() const { return m\_bb; }

返回标志所在矩形区域的信息

inline bool Tracker::IsInitialised() const { return m\_initialised; }

返回Tracker是否被初始化

void Tracker::UpdateLearner(const ImageRep& image)

升级Tracker中的信息，为下次更高效追踪提供数据和样本

###### Common

**1. ICommand.hpp**

**1.1 ICommandParam类（纯虚类）**

这个类主要负责ICommand中的参数的处理。

成员函数：

ICommandParam() {}

构造函数

virtual ~ICommandParam() {}

析构函数

virtual int GetParamNum()=0;

获取参数个数

virtual void\* GetParam(int index)=0;

参数index表示第几个参数，这个函数负责获取第index个参数的内容

**1.2 ICommand 类（纯虚类）**

主要负责根据view层中触发的事件（比如按了某个按钮），进行对应的命令操作

成员函数：

ICommand() {}

构造函数

virtual ~ICommand() {}

析构函数

virtual void Execute(const boost::shared\_ptr<ICommandParam> &param)=0;

执行命令函数，参数传入由ICommandParam处理后过的对应的参数数据

**2. INotification.hpp**

**2.1 INotification类（纯虚类）**

成员函数：

INotification(){}

构造函数

virtual ~INotification(){}

析构函数

virtual void OnPropertyChanged(const std::string &property)=0;

这个函数负责当Model层数据取出处理好之后给予View层一个显示的信号

**3.OpenCommandParam.hpp**

**3.1 OpenCommandParam类，继承自ICommandParam 类**

负责记录Open这个命令的参数信息

成员变量：

std::string video\_filename

video\_filename变量记录录像名

std::string mark\_filename;

mark\_filename变量记录标记好的信息

成员函数：

OpenCommandParam() {}

构造函数

~OpenCommandParam() {}

析构函数

int GetParamNum() { return 1; }

获取参数数量

void SetParam(const std::string &s) { video\_filename=s; }

获取参数（即外部获取的录像路径名）传入类内部交给video\_filename保存

void\* GetParam(int index)

获取第index个参数的内容

###### Model

**1.TrafficSignModel .hpp**

**1.1 TrafficSignModel类**

主要用于Model层中负责与数据交换，包括保存、提取和修改数据。

成员变量：

boost::shared\_ptr<ImageData> sp\_image

智能指针sp\_image来记录从保存的数据中取出来的图片信息

boost::shared\_ptr<VideoData> sp\_video

智能指针sp\_video来记录录像信息

boost::shared\_ptr<MarkData> sp\_sMark, sp\_tMark

智能指针sp\_Mark来记录图像中的标记信息

boost::shared\_ptr<Config> sp\_config

智能指针sp\_config来记录配置文件的信息

std::string videoName

记录录像名的变量videoName

成员函数：

TrafficSignModel()

构造函数

~TrafficSignModel()

析构函数

void OpenVideo(const std::string &filename)

打开指定的录像

void SetImage(const cv::Mat &image)

将传入的图像信息复制保存到数据中

void ReadMark(const std::string &filename)

读取录像中的标记信息

boost::shared\_ptr<ImageData> GetImage()

读取保存数据中的图像信息，返回的是指向Mat图像的智能指针

boost::shared\_ptr<Config> GetConfig()

读取配置文件信息，返回的是指向配置文件信息的智能指针

boost::shared\_ptr<MarkData> GetMark(int)

获取标记信息，返回的是指向标记信息的智能指针

cv::Mat \*GetFrame(int frameInd)

获取录像的某一帧

const std::string &GetVideoName()

获取录像的名字

###### Utility

**1.ImageData.hpp**

**1.1 ImageData类**

主要负责存放读取修改图像信息

成员变量：

cv::Mat image

存放图像信息

成员函数：

cv::Mat \* GetImage()

读取图像信息，以指针形式返回

void SetImage(const cv::Mat & image)

读入图像信息，并复制信息保存入对象中

**2 VideoDate.hpp**

**2.1 VideoDate类**

主要功能是视频的读取存储和视频帧画面的提取输出。

成员变量

cv::VideoCapture cap

用于获取视频对象，以及对视频处理等功能。

cv::Mat frame

用于储存视频的某一帧的图像。

int currentFrame, totFrame

当前帧数和视频总帧数。

成员函数

void OpenVideo(const std::string &)

利用VideoCapture的open()成员函数打开指定目录的视频文件。

cv::Mat \*GetFrame()，cv::Mat \*GetFrame(int)

获取指定帧（未指定则为当前帧）画面并输出。

###### ViewModel

**1. TraficSignViewModel类**

成员变量：

boost::shared\_ptr<ICommand> sp\_OpenCommand

打开视频命令指针

boost::shared\_ptr<ICommand> sp\_RunCommand

捕获交通标志命令指针

boost::shared\_ptr<ICommand> sp\_ReadCommand

读取标记好的视频图像命令指针

boost::shared\_ptr<TrafficSignModel> sp\_Model

Model层对象指针

boost::shared\_ptr<INotification> event

事件标记指针，用来通知View层各种数据是否已准备好

boost::shared\_ptr<QImage> sp\_image\_view, sp\_mark\_view

用于储存原视频图像和标记好的视频图像

成员函数

TrafficSignViewModel()，

构造函数，用于各指针的初始化。

boost::shared\_ptr<ICommand> GetOpenCommand()

发出打开视频命令信号。

boost::shared\_ptr<ICommand> GetRunCommand()

发出捕获交通标志命令信号。

boost::shared\_ptr<ICommand> GetReadCommand()

发出读取标记好的图像命令信号。

boost::shared\_ptr<QImage> GetImagePtr(const int &)

将图像指针（原图或标记好的图）发送到View层。

void SetEvent(const boost::shared\_ptr<INotification> &e)

设定事件完成标记。

void SetModel(boost::shared\_ptr<TrafficSignModel> &model)

读取Model层对象。

void OpenVideo(const std::string &filename)

向Model层发出打开视频命令。

void TrackSign(const std::string &filename)

向Model层发出捕获交通标志命令。

void ReadMark(const std::string &filename)

向Model层发出读取已标记好的图像命令。

**2. OpenCommand类（ICommand的子类，ReadCommmad/RunCommand类似）**

主要用于存储打开视频这个命令。令，对Model层发出指令请求。

成员变量

TrafficSignViewModel \*m\_ViewModel

ViewModel层对象。

成员函数

OpenCommand(TrafficSignViewModel \*m)

构造函数。

~OpenCommand()

析构函数。

void Execute(const boost::shared\_ptr<ICommandParam> &param)

向ViewModel层传入打开视频命令。

###### View

**1. Traffic\_signMainWindow类**

主要用于对界面鼠标点击事件的捕获和图像的显示。

成员变量

boost::shared\_ptr<ICommand> sp\_OpenCommand

打开视频命令指针

boost::shared\_ptr<ICommand> sp\_RunCommand

捕获交通标志命令指针

boost::shared\_ptr<ICommand> sp\_ReadCommand

读取标记好的视频图像命令指针

boost::shared\_ptr<QImage> sp\_image\_view, sp\_mark\_view

用于储存原视频图像和标记好的视频图像

boost::shared\_ptr<INotification> sp\_Event

事件标记指针，用来通知View层各种数据是否已准备好

成员函数

void on\_VideoButton\_clicked();

void on\_ExitButton\_clicked();

void on\_RunButton\_clicked();

用于捕获和触发鼠标单击事件

QTimer \*running\_timer

QT中断

boost::shared\_ptr<QImage> GetImage(const int &)

获取要显示的图像

boost::shared\_ptr<INotification> GetEvent()

从ViewModel获取信息准备情况的通知。

explicit Traffic\_signMainWindow(QWidget \*parent = 0)

构造函数。

~Traffic\_signMainWindow()

析构函数。

void SetOpenCommand(const boost::shared\_ptr<ICommand> &ptr);

void SetReadCommand(const boost::shared\_ptr<ICommand> &ptr);

void SetRunCommand(const boost::shared\_ptr<ICommand> &ptr);

向ViewModel层发送打开视频/开始捕获/读取结果命令

void SetImage(const boost::shared\_ptr<QImage> &ptr);

void SetMark(const boost::shared\_ptr<QImage> &ptr);

储存图像信息。

QLabel \*GetLabel(const std::string &)

获取当前图像的性质（原图或标志图）。

**2. MainWindowSink类（INotification的子类）**

主要用于当前显示属性更改的通知。

成员变量

Traffic\_signMainWindow \*p\_Window

主窗口对象指针。

成员函数

MainWindowSink(Traffic\_signMainWindow \*p):p\_Window(p)

构造函数初始化指针。

void OnPropertyChanged(const std::string &property)

标记当前播放为视频或标志或已经捕获完毕。

## 开发日志

**7月5日：**确定课题，确定开发环境和依赖库，大家一起在自己的计算机上配置开发环境。遇到的主要问题：在虚拟机中安装Ubuntu 14.04 Desktop比较慢；Windows 8系统下的VMWare WorkStation不太稳定，安装过程中常常卡住；Mac OS X系统下的Parallels Desktop 9对Ubuntu 14.04 Desktop兼容性不好，安装完成后分辨率有问题，只能换用VMWare Fusion。

李逸婷写了一份GitHub使用的简要说明并push到我们的repo供组员们参考。

**7月6日：**继续安装和配置开发环境。确定大致分工并且根据自己的任务进行资料的查阅和学习，其中：

高涛负责前端图形界面的绘制和开发，系统架构；

李逸婷负责图像读写、图像信息处理相关工作；

毛晨炀负责标记框的绘制、结果的保存；

徐超颖负责视频处理相关工作；

徐可添负责系统架构和追踪算法相关工作。

遇到的困难有：大家都是第一次接触MVVM框架，不能很好理解其设计思想和设计方法，决定由高涛和徐可添重点学习，由高涛负责View层、ViewModel层，徐可添负责ViewModel层、Model层的方式来搭建框架。

将课题信息和分工等push到GitHub上。

**7月7日：**环境基本都配置好了。高涛和徐可添参考第一组的程序框架，进行大体框架的搭建，但对于INotification类的作用不能很好理解。小组讨论决定，先完成一个具有“按下一个按钮给出一个弹出窗口供用户选择路径，然后读入一个视频并显示其第一帧”这样基本功能的程序，作为第一轮迭代的目标。

大家都在自己的计算机上完成了利用OpenCV读入一张图片并显示的小程序，确保开发环境正常。

**7月8日：**继续前一天的工作。徐超颖、毛晨炀的系统在更新后驱动程序坏了，无法联网，只能重新从头安装。组长学习了Qt Creator的使用，并和组员分享。

**7月9日：**白天组长去社会实践了…

晚上完成了大体框架的搭建。李逸婷、徐超颖学习并实现了几个利用OpenCV进行图像、视频的输入输出的类。毛晨炀学习了利用OpenCV绘制标记框、对框定区域保存的方法。高涛完成了前端界面，徐可添将这些类和界面整合进之前完成的MVVM框架。能够基本达到预期效果，但应用程序第一次读入的视频无法正常显示第一帧，之后读入的视频都能正常显示。

**7月10日：**我们听取了袁老师对第一个程序的指导意见，明白了View和ViewModel之间的正确关系，明白了之前贪图方便将ViewModel全部暴露给View的做法的缺陷，理解了INotification的工作机制，并对前一天完成的程序做了相关修改。第一轮迭代完成。

小组成员参考网上《CMake实践》一文，集体系统学习了CMake的工作原理。

**7月11日：**修正原来程序中的一些bug。

我们使用的追踪算法基于ICCV 2011的“Struck: Structured Output Tracking with Kernels”一文，原作者提供的源代码使用Makefile来管理工程，李逸婷学习了Makefile并尝试编译链接，但是静态库链接始终有问题，最终还是通过新写一个cmakelists.txt解决了问题。我们用作者提供的样例尝试了追踪程序。

组长开始结合论文阅读Struck程序的源代码，并尝试将其整合进我们的程序，作为第二轮迭代的目标。

**7月12日：**因机房有考试停课一天。

为实现对视频的持续处理，组长学习Qt中断机制，成功将Struck程序整合进我们的程序并完成相关数据传递、消息传递的逻辑。高涛对界面作出相应的修改，完善View层逻辑。完成第二轮迭代。其他组员对程序功能进行测试，提交了一些bug，其中最大的问题是：在配置文件中设置视频缩放可能会引起崩溃。

**7月13日：**向袁老师展示程序功能。

修正bug。增加了独立显示标定窗口内容的功能，完善了页面布局。完成第三轮迭代。小组成员共同撰写实验报告。其中：

徐可添负责摘要和排版

高涛负责设计结构和整合文档

徐超颖、李逸婷负责主要实现

毛晨炀负责程序运行

开发日志为共同完成。

**7月14日至21日：**这一周进行了收尾工作。进行的主要工作有：发现了视频读取时的一个bug，造成了整个系统的性能瓶颈，修正后系统的运行速度有很大的提高。修改了Struck算法，使之能够追踪大小变化的交通标志。至此，完成了第四轮迭代。

此外，组长和组员各自撰写了自己的报告。组长修改了总报告。

# 程序运行

## 程序使用说明

编译程序可以参考摘要中“编译运行”一小节。

在开始运行前，请确保可执行文件所在文件夹中有配置文件config.txt。该文件的样例可以在项目的config文件夹中找到，支持对视频进行缩放、选择追踪算法使用的图像特征等。

运行前，先点击video按钮选择一个视频，然后点击run按钮导入被追踪对象的信息并开始追踪。追踪对象的信息满足以下格式：

|  |
| --- |
| start\_frame start\_lx start\_ly start\_rx start\_ry  end\_frame end\_lx end\_ly end\_rx end\_ry |

分别表示：初始帧、待追踪区域左上角坐标x和y、待追踪区域右下角坐标x和y、追踪结束帧、结束时待追踪区域左上角坐标x和y、右下角坐标x和y。

## 程序运行示例

以33489.mp4为例，展示运行结果：

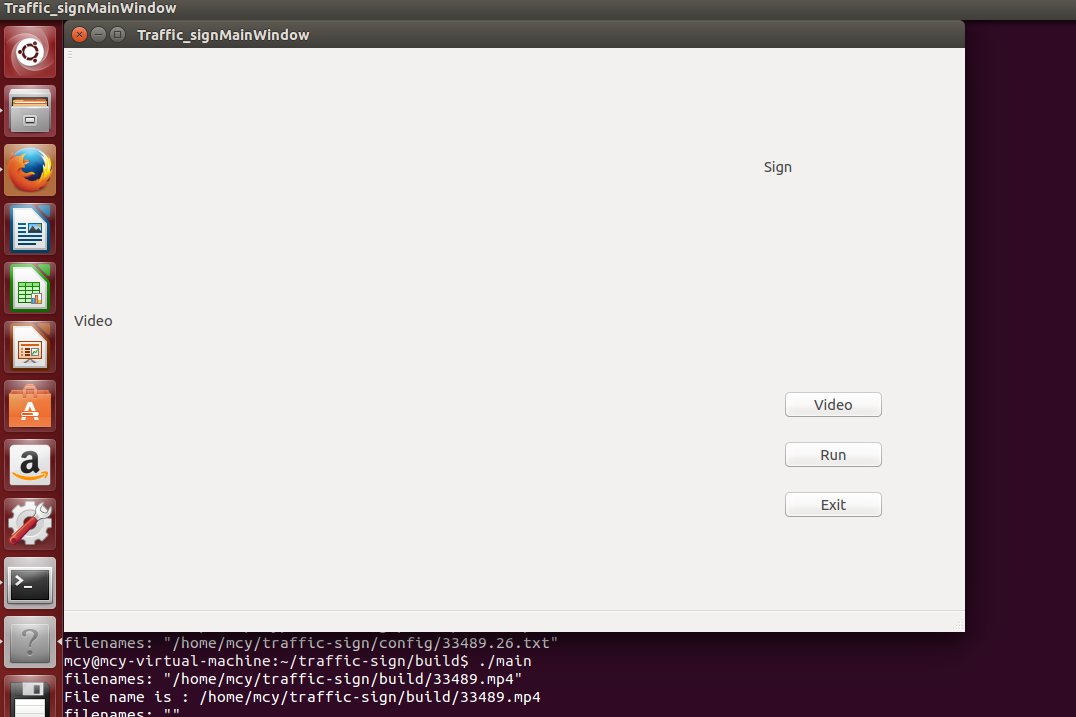


图 3主界面

载入视频，按video按钮：

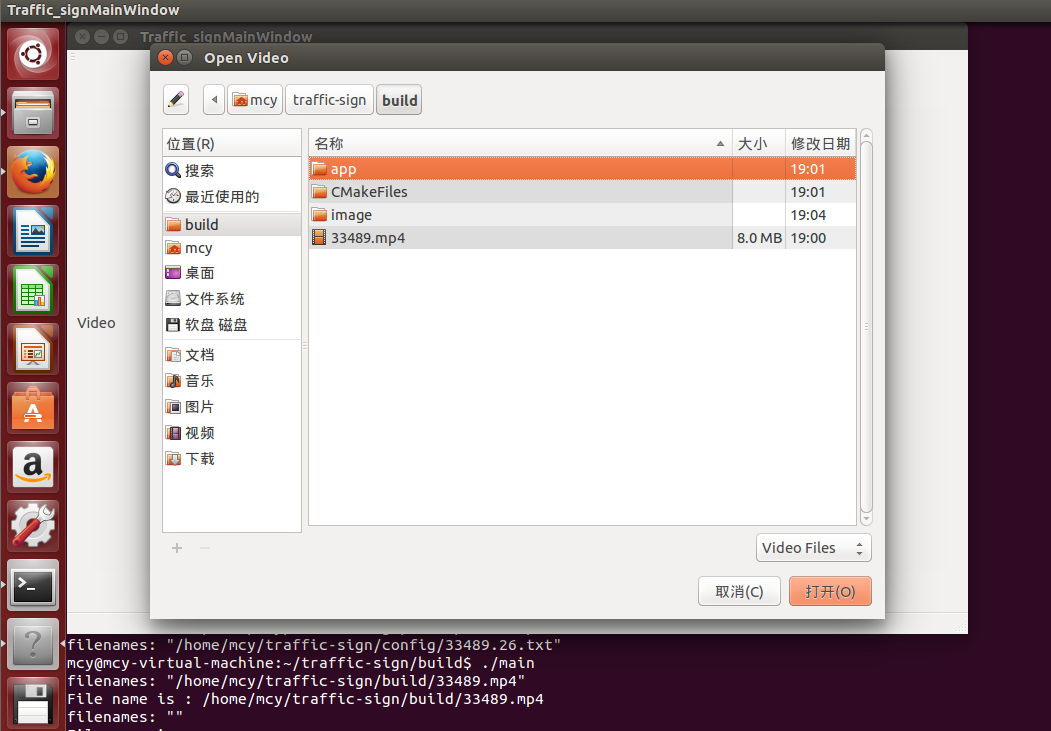


图 4

载入33489.mp4视频文件

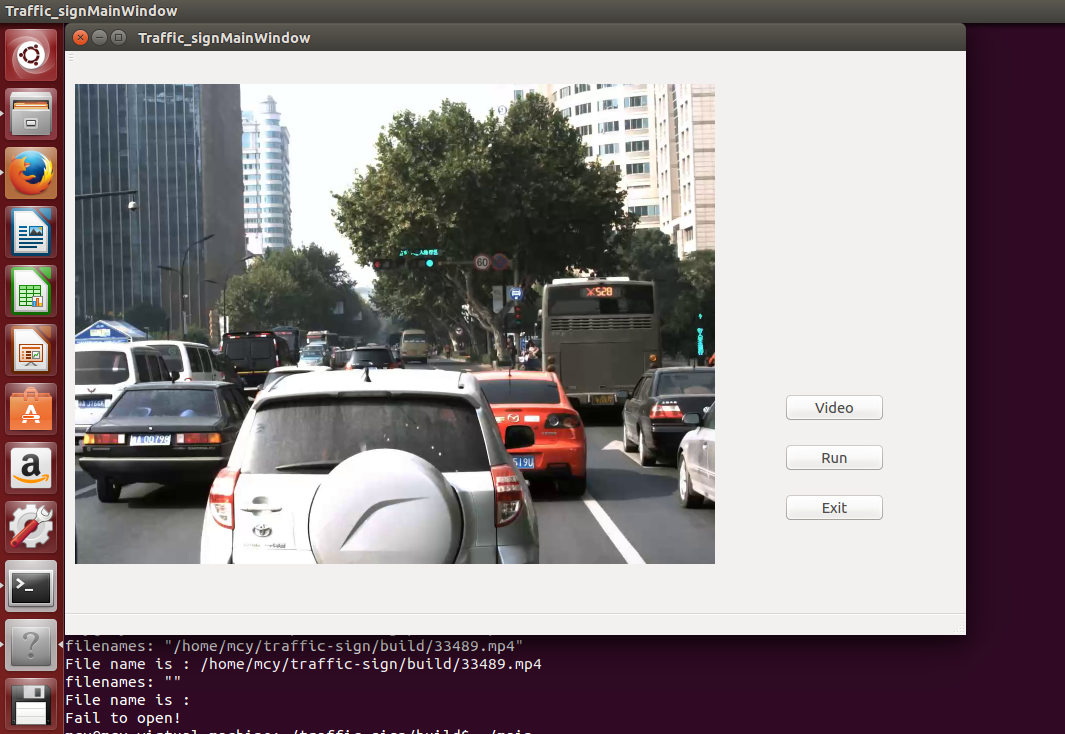


图 5

载入配置文件，按run按钮：

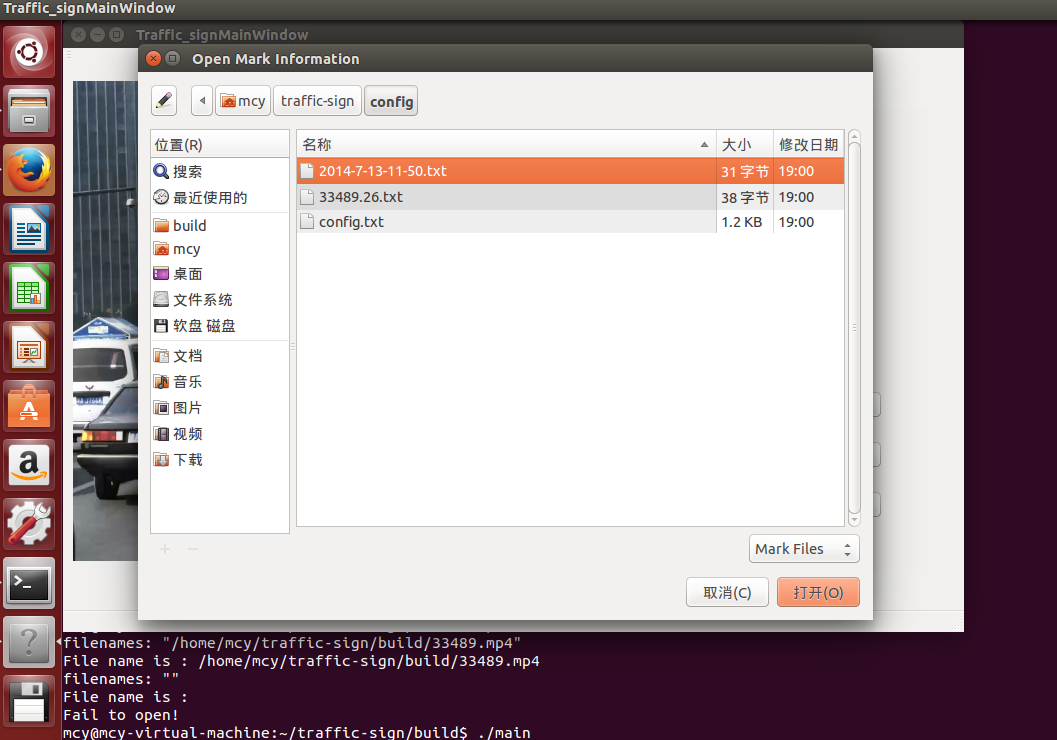


图 6

载入333489.txt后运行：

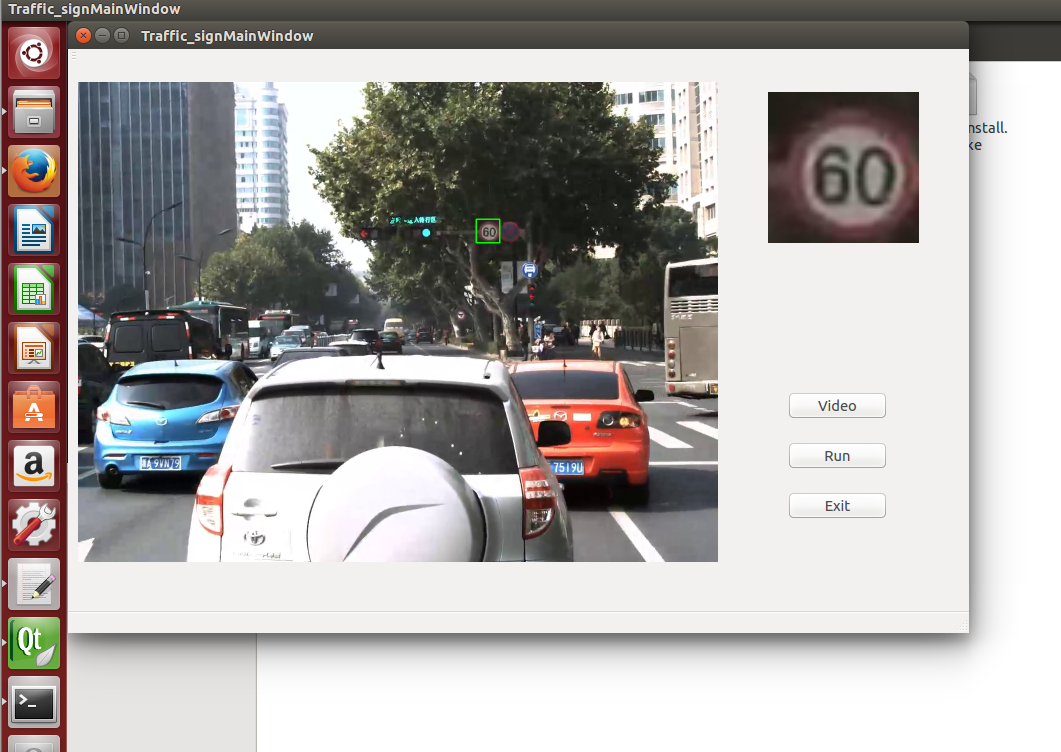


图 7

框在随标识的远近变化

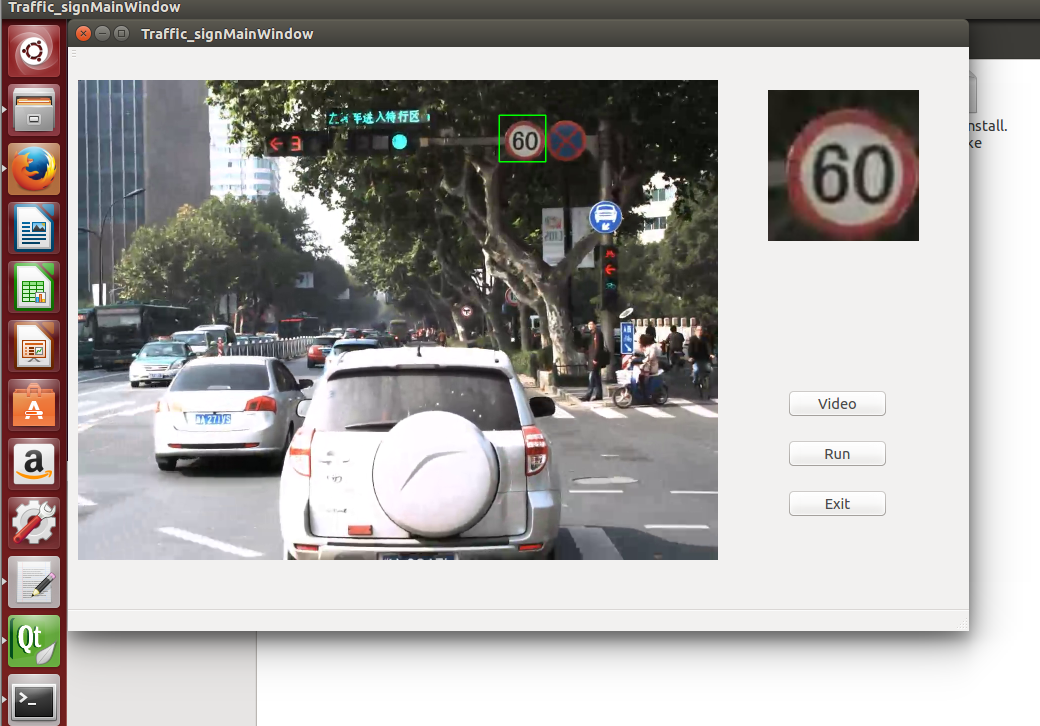


图 8

换配置文件，标识为绿灯

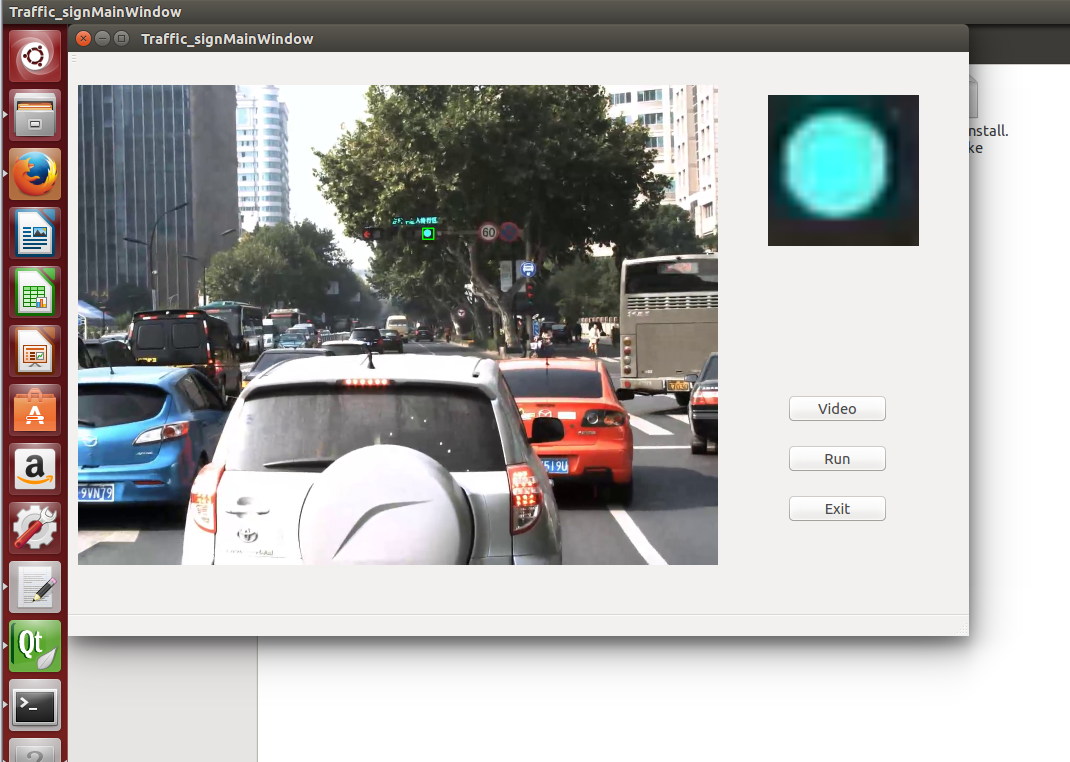


图 9

按Exit按钮退出：

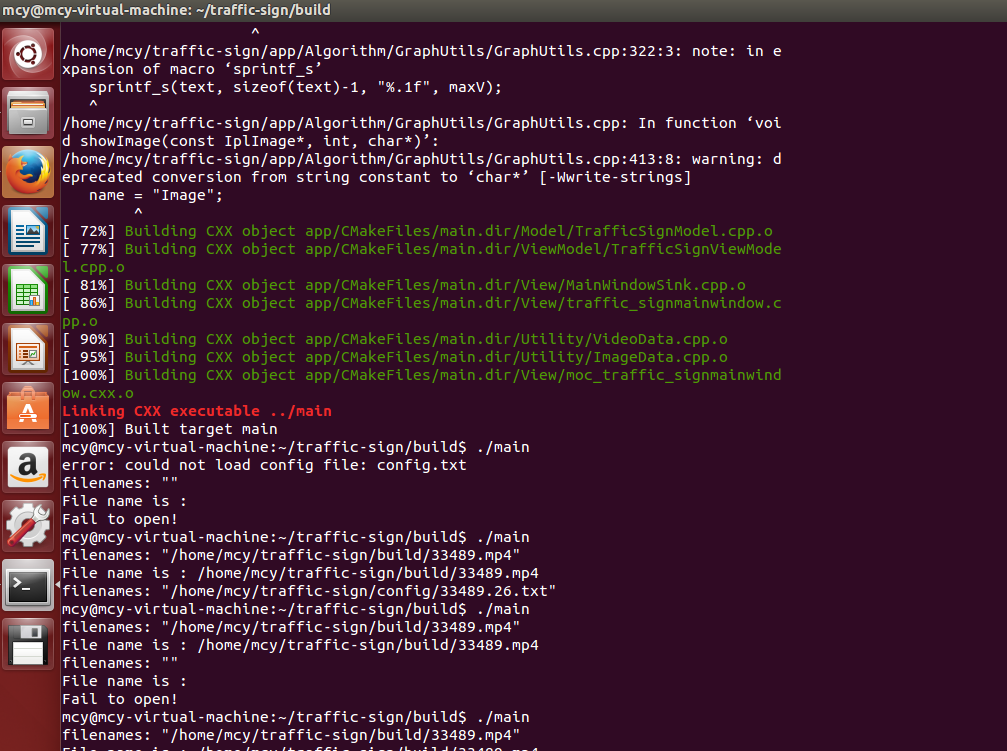


图 10

在image文件中记录下了交通标志的图片文件：

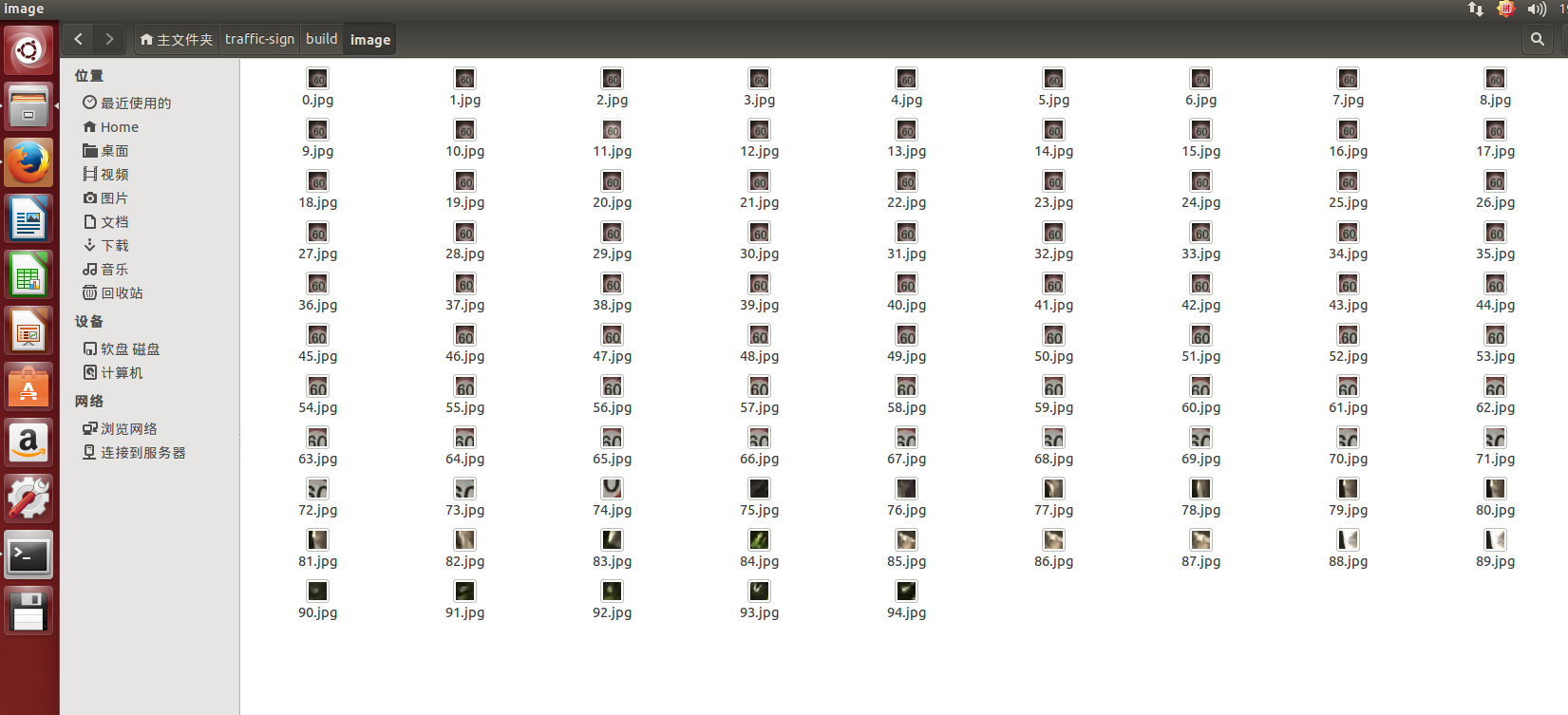


图 11

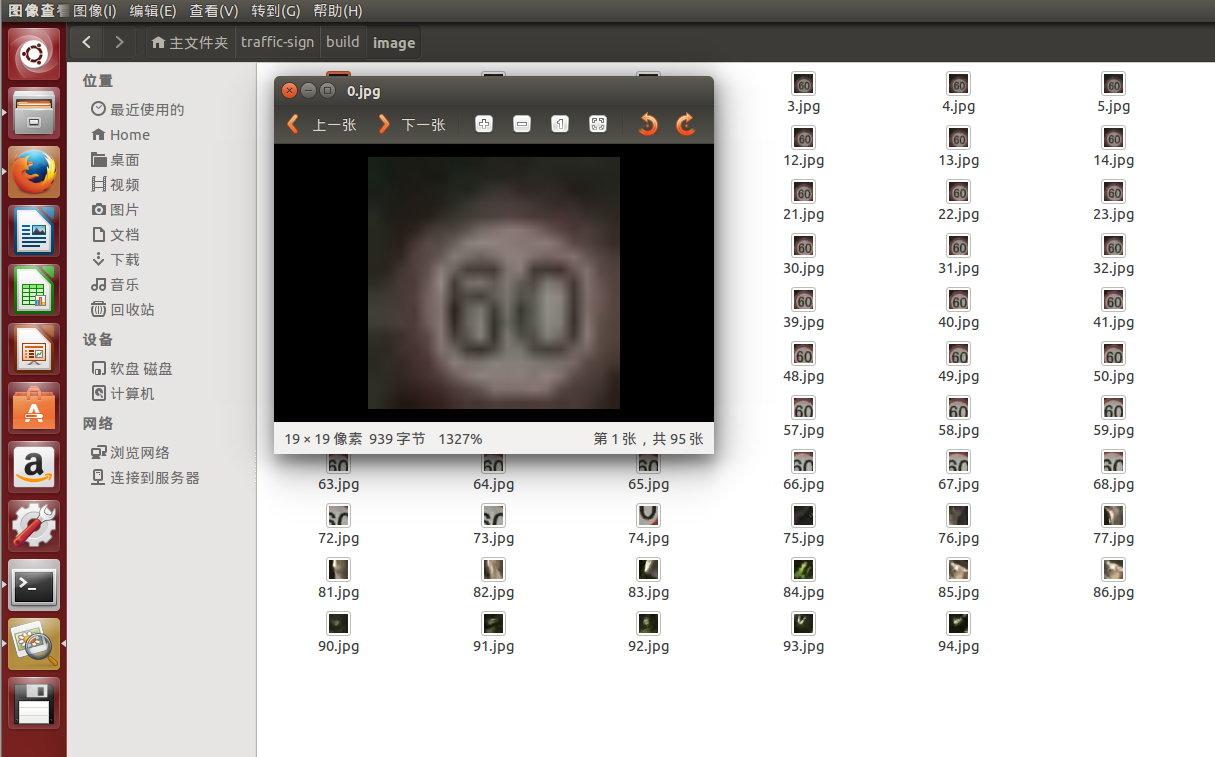


图 12

# 附件（GitHub地址）

<https://github.com/Zhejiang-University-GKC/traffic-sign>

2014年7月21日