结题报告

《视频中移动车辆的定位和车牌 提取系统的设计与实现》

**指导教师：华保健**

[摘 要 2](#_Toc30055)

[第一章 常用的车牌定位方法介绍 3](#_Toc3880)

[1.基于边缘检测的车牌定位系统 4](#_Toc3562)

[2.基于水平跳点的车牌定位系统 4](#_Toc11780)

[3.基于颜色的车牌定位系统 4](#_Toc14149)

[第二章 功能的实现 4](#_Toc25281)

[2.1 使用OpenCV中的功能 5](#_Toc16065)

[2.1.1图像的存取 5](#_Toc13019)

[2.1.2图像大小的转换 5](#_Toc5105)

[2.1.3图像格式的转换 6](#_Toc30581)

[2.1.4视频的读取 6](#_Toc15298)

[2.1.5图像平滑 6](#_Toc22716)

[2.2 Qt基础 7](#_Toc24798)

[2.3 系统设计与实现 7](#_Toc5428)

[2.3.1系统车辆定位模块的实现 8](#_Toc5169)

[2.3.2系统车牌定位模块的实现 9](#_Toc31917)

[2.4系统关键技术的实现 11](#_Toc949)

[2.4.1 高斯混合模型的建立与更新 11](#_Toc3319)

[2.4.2 顶帽变换的实现 12](#_Toc15297)

[2.4.3 能量滤波的实现 13](#_Toc2662)

[4.4.4 Opencv内部的图像结构与Qt内部图像结构的转换 14](#_Toc7517)

# 摘 要

在现阶段的交通执法领域中，交警在记录交通车辆信息，判定车辆是否违章的工作中，投入了大量的人力物力。利用视频技术检测和定位汽车牌照是从技术上实现智能交通管理的重要途径，可以有效防控交通违章和交通事故。随着交通的发展，我们对智能化交通的需求，正变得越来越强烈。为节约成本，我们需要通过交通监控系统提供适时地的车辆图片，对获得的图像进行分析，获得车辆的牌照信息，进行智能化的判定，来完成对交通车辆的实时管理。

针对这一现象，本文中介绍了实现从视频中提取运动的车辆，以及，对得到的车辆进行车牌的定位等工作。

本文主要工的作内容如下：

1. 学习数字图像处理基础，掌握图像的基本格式、编码和颜色空间等知识。调研文献，整理车牌检测和定位相关方法；

2. 学习C++、OpenCV以及Qt的开发环境，掌握了将三者结合，使用QC++配合QpenCV库，对图像进行读取，存储以及对图像进行处理的工作，最终形成一整套的系统体系。

3. 调查了一系列的视频中车辆定位，以及车牌定位的方法，对高斯背景建模、背景差分、能量滤波等一系列算法进行综合运用，并添加了高帽变换等对系统进行改进，设计实现了一种较好的车辆定位以及车牌定位的系统，经过大量的实验表明，本文实现的算法具有较好的效果。

## 

## 

## 第一章 常用的车牌定位方法介绍

车牌检测是进行车牌是别的基础，直接影响到系统的稳定性和准确率，因此，现阶段，提出了多种方法来进行车牌定位。

### 1.基于边缘检测的车牌定位系统

由于车辆牌照区域颜色变化明显，通过对图像进行边缘检测，能够清楚地得到车牌区域的边缘信息[6]，通常，经过Canny边缘检测之后，在车牌的区域中能够得到明显的矩形边缘。下一步，通过形态学的判断，在图像中寻找矩形区域来得到车牌的区域。

该算法的优点是算法简单易懂，对车牌边缘检测之后，能够明显的将车牌区域显示出来。但是由于车牌区域的情况较为复杂，边缘检测的效果很容易受到车牌环境的影响，而且在寻找矩形的过程中，容易背图像中其他的矩形框干扰，鲁棒性差。

### 2.基于水平跳点的车牌定位系统

该算法的理论基础是灰度化后的车牌部分，横向像素的跳跃现象明显，并且跳跃的次数稳定在一个区间内，因此通过寻找水平跳跃点集中的区域，就可以将车牌区域寻找出来。

该算法的优点是算法简单易懂，主要是根据先验知识进行判定，但是，由于一些车辆的颜色图案并不单一，因此，该方法中，车牌占图像的比例越小，效果越差。

### 3.基于颜色的车牌定位系统

该算法的理论基础车牌的颜色较为固定，主要的车牌有蓝色、黄色、黑色和白色四种颜色，因此，我们只要在车辆图像中寻找到这几种颜色集中的区域，就可以找到车牌。

该算法优点是适用性较广，但是，对于车辆颜色与车牌颜色相似的情况，比较难以解决；同时，单纯的判断图像中的RGB信息，容易受到光线的干扰，难以达到预期的效果。

## 第二章 功能的实现

对于视屏中移动车辆的定位以及静态图像中车牌定位的实现，主要算法的设计使用了openCV函数与Qt内置函数。极大的提高了系统的运行的效率，并且可以多平台的运行。

本章给出了该系统的设计过程和系统的实现过程，并使用了大量的实验对系统的鲁棒性进行测试。

## 2.1 使用OpenCV中的功能

OpenCV的库函数中封装了对数字图像的常用操作。其目的是创建一个简单实用的计算机视觉框架，能够方便系统开发者，同时也能够促进计算机视觉系统的设计、应用和发展。

在OpenCV中，封装了超过500个基本的库函数。涵盖了图像操作的各个领域方面。

在本系统中，主要使用了包括着图像的存取，视频的读取，图像的显示，图像的平滑，高斯背景建模，图像的膨胀与腐蚀等基本操作。

### 2.1.1图像的存取

cvLoadImage()函数是OpenCV中封装的图像读取函数。cvLoadImage()函数的声明如下：

IplImage\* cvLoadImage ( const char \* filename,

int iscolor=CV\_LOAD\_IMAGE\_COLOR );

由于图像存储的过程中，其文件的头几个字符包含着文件的格式信息，因此，在读入图像时，只需要读取其文件头，就可以确定文件的编码方式。第二个参数有一下几个值可以选择：

CV\_LOAD\_IMAGE\_ANYDEPTH：读入任意深度的图像，输入图像格式可以为8位或者16位的无符号数，也可以为32位有符号或者32位浮点型数。；

CV\_LOAD\_IMAGE\_COLOR：默认原图为3通道图像读入；

CV\_LOAD\_IMAGE\_GRAYSCALE：默认原图为单通道读入；

CV\_LOAD\_IMAGE\_UNCHANGED：读入原始图像 。

int iscolor的默认值为CV\_LOAD\_IMAGE\_COLOR

### 2.1.2图像大小的转换

OpenCV中，虽然提供了独立的显示窗口机制，由于，系统使用Qt框架，要将图像显示在Qt窗口中，需要做一定的转换。

首先，需要将原始图像，装换成一定大小的图像，OpenCV中 中提供了cvResize()函数。cvResize()函数声明如下：

cvResize(const CvArr \*src, //待处理图像

CvArr \*dst, //处理完成后的图像

int interpolation = CV\_INTER\_LINEAR //使用的转换方法

);

在对图像大小的转换中，使用不同的方法，其转换的效率以及效果会有差异，系统开发者可以根据不同的情况，选择不同的方法，如下所示。

CV\_INTER\_LINEAR: 函数的默认选择，方法为双线性插值；

CV\_INTER\_NN: 方法为最近邻居插补；

CV\_INTER\_AREA :像素面积相关重采样；

CV\_INTER\_CUBIC: 方法为三次卷积法。

### 2.1.3图像格式的转换

cvCvtColor()是OpevCV提供的图像类型的转换函数;此函数能够进行几乎所有的颜色空间之间的相互转换，函数的定义如下：

cvCvtColor( const CvArr \* src, //原图像

CvArr \* dst, //处理完后的图像

Int code); //处理方式

Code有十分丰富的取值，其中常用的取值如下所示。

CV\_BGR2HSV：将RGB颜色空间转换为HSV颜色空间；

CV\_BGR2GRAY：将RGB颜色空间转换为Gray灰度图像；

CV\_BGR2YCrCb：将RGB颜色空间转换为YCrCb颜色空间；

CV\_RGB2YUV：将RGB颜色空间转换为YUV颜色空间；

CV\_RGB2GRAY：将Gray灰度图转换为RGB颜色空间；

### 2.1.4视频的读取

在OpenCV中，提供了视频读取的函数cvCaptureFromFile()，该函数能够自动读取视频文件头，获得视屏的类型，在根据视频的编码格式，将视频读入系统之中。该函数的声明如下。

CvCapture\* pCapture =cvCaptureFromFile(const char \*filename);

Filename即为视屏存放的地址，只需要将文件的绝对路径放入，就能从将视频读取出来。

### 

### 2.1.5图像平滑

在图像处理中，图像的平滑占有极为重要的作用，图像的平滑，能够减少噪声对图像的干扰，提高处理效果。

Void cvSmooth (

Const CvArr \* src, //待处理图像

CvArr\* dst, //处理完后的图像

int smoothType= CV\_GAUSSIAN,

int param1 = 3,

int param2 = 0,

double param3 = 0,

double param4 = 0,

);

cvSmooth()包含4个参数：param1，param2，param3和param4。这些参数的含义取决于参数smoothType的取值。smoothType的取值有如下5种：

CV\_BLUR：简单模糊，对每个像素的param1×param2邻域求和，并作缩放1/( param1×param2);

CV\_MEDIAN：中值模糊，对图像进行窗口大小为param1×param2的中值滤波。

CV\_BLUR\_NO\_SCANLE：无缩放模糊，对每个像素的param1×param2邻域求和。

CV\_GAUSSIAN：高斯模糊，对图像进行窗口大小为param1×param2的高斯卷积。

CV\_BILATERAL：双边滤波，应用双线性3×3滤波，颜色sigma=param1，空间sigma=param2。

## 2.2 Qt基础

Qt能够支持包括Microsoft Windows 95/98， Microsoft Windows NT， Linux， Solaris， SunOS， HP-UX， Digital UNIX (OSF/1， Tru64)， Irix， FreeBSD，等多种平台。并且Qt 的模块化设计十分成熟，编程者使用方便，并且提供了十分具有特色的signals/slots机制。这使得各个元件 之间的协同工作变得十分简单。

而且,Qt有着丰富的API，其提供了超过250个涵盖各领域的C和C++函数，极大地提高了编程的效率。

## 2.3 系统设计与实现

该系统，主要分为两大模块，第一模块是视频中运动车辆的定位[18]，第二模块式静态图像中车牌的定位[19]。

### 2.3.1系统车辆定位模块的实现

第一模块运动车辆定位主要有三部分组成，分别是视频的输入，图像的运算，图像的输出。如图2.1所示。

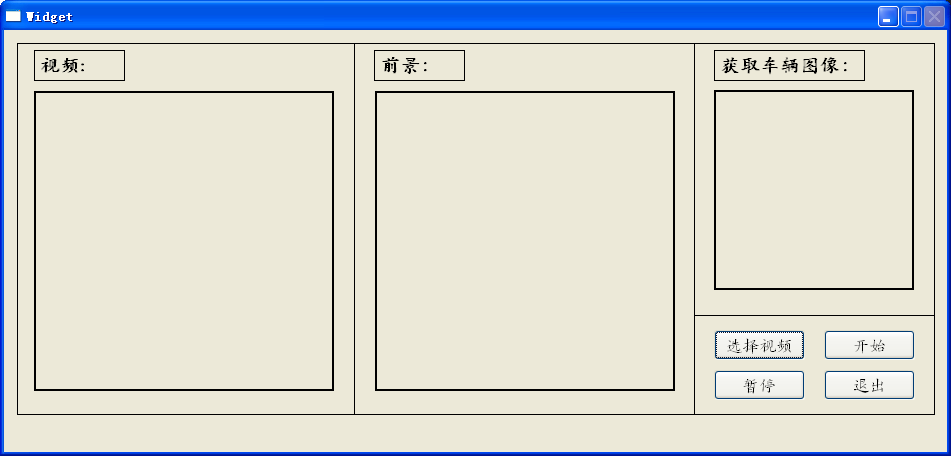


图2.1 运动中车辆定位系统窗口

如图2.1所示，该系统有四个按钮，分别为选择视频按钮，开始按钮，暂停及退出按钮。在运行系统之前，首先要使用选择视频按钮获得视频的绝对路径，如图2.2所示

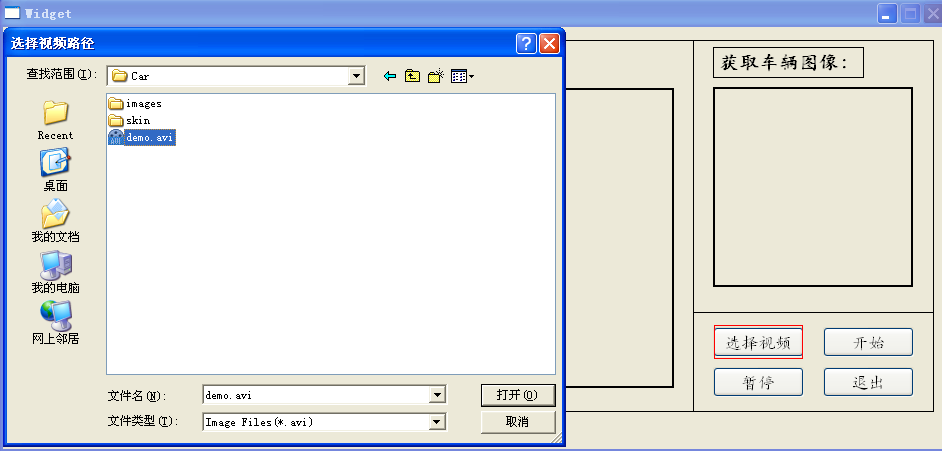


图2.2 确定视频路径窗口

当选择了视频的路径，就可以进行播放视屏，以及获得运动的车辆图像，只要点击开始按钮即可。系统运行时，会自动计算，并将前景图和获得的车辆图像显示到窗口上，其系统运行的结构如图2.3所示。



图2.3 系统运行时的窗口

在该系统模块中，主要的成员函数如下。

class Widget : public QWidget

{

public:

explicit Widget(QWidget \*parent = 0);

//读取视频，并进行运算

void ReadAvi();

//在视频前景图中寻找运动中的车辆

void FindCar(IplImage \*CarImg,IplImage \*tmpFrame);

private slots:

//选择视频按钮的响应函数

void on\_Button\_OpenVideo\_clicked();

//开始按钮的响应函数

void on\_Button\_Start\_clicked();

//暂停按钮的响应函数

void on\_Button\_Stop\_clicked();

//退出按钮的响应函数

void on\_Button\_Exit\_clicked();

};

### 2.3.2系统车牌定位模块的实现

第二模块静态图像中车牌的定位主要由三部分组成，分别是车牌图片的输入，算法的运行[20]和结果的输出。在该系统中，对每一个关键环节，都提供了结果输出系统，使用者可以清晰的看到，结果是如何一步一步的得到的。其系统演示如图2.4所示。

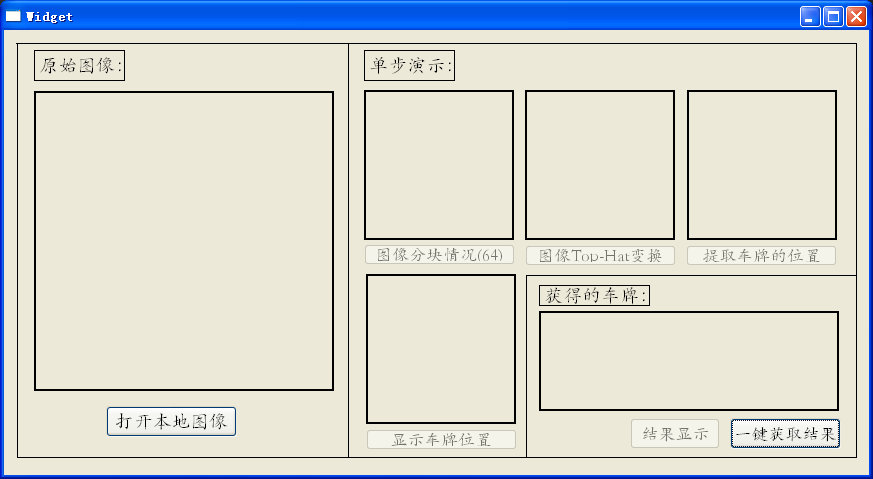


图2.4 系统运行时的窗口

在该系统中，包括逐步取得运行结果和一键获得运行结果两种方式，当用户输入图像后，点击单步演示下的各个按钮，会获得各个步骤的计算结构。如图2.5所示。



图2.4 系统单步演示运行结果

## 当用户选择一键显示的按钮，系统也会自动的将各个单步的演示结果显示出来，便于用户操作，其系统的运行结果如图2.6所示。



图2.4 系统一键演示运行结果

## 2.4系统关键技术的实现

### 2.4.1 高斯混合模型的建立与更新

在本系统中，高斯背景建模具有十分重要的作用，直接关系到运动车辆的提取，本系统主要利用OpenCV函数进行高斯混合模型的建立和更新[21]，其主要代码以及注释如下所示。

//初始化参数

CvGaussBGStatModelParams\* params = new CvGaussBGStatModelParams;

params->win\_size = 50;

params->n\_gauss = 3;

params->bg\_threshold = 0.7;

params->std\_threshold = 3.5;

params->minArea = 15;

params->weight\_init = 0.033;

params->variance\_init = 30;

//打开视频文件

pCapture = cvCaptureFromFile(VideoPath.toStdString().c\_str());

//初始化高斯混合模型参数

CvGaussBGModel\* bg\_model=NULL;

//视频的第一帧进入

if(nFrmNum == 1)

{

pBkImg = cvCreateImage(cvSize(pFrame->width, pFrame->height), IPL\_DEPTH\_8U,3);

pFrImg = cvCreateImage(cvSize(pFrame->width, pFrame->height), IPL\_DEPTH\_8U,1);

//高斯背景建模，pFrame可以是多通道图像也可以是单通道图像

//cvCreateGaussianBGModel函数返回值为CvBGStatModel\*，

//需要强制转换成CvGaussBGModel\*

bg\_model= (CvGaussBGModel\*)cvCreateGaussianBGModel(pFrame,params);

} else

{

//非第一帧视频进入

//更新高斯模型

cvUpdateBGStatModel(pFrame, (CvBGStatModel \*)bg\_model );

//pFrImg为前景图像，只能为单通道

//pBkImg为背景图像，可以为单通道或与pFrame通道数相同

cvCopy(bg\_model->foreground,pFrImg,0);

cvCopy(bg\_model->background,pBkImg,0);

//把图像正过来

pBkImg->origin=1;

}

}

### 2.4.2 顶帽变换的实现

顶帽变换是处理静态图片的关键步骤，能够将车牌图片的复杂背景去除。使得我们在一个较为单一背景的图像中使用能量滤波，寻找车牌，极大的提高了寻找车牌的准确率和效率。同样的该系统使用OpenCV中所提供的函数来实现。其具体代码及注释如下所示。

//顶帽变换函数，DirImg为原图像，Timage3为目标图像，tmp为缓存图像

cvMorphologyEx(**DirImage**,Timage3,tmp,NULL,CV\_MOP\_TOPHAT,1);

//赋值高帽变换后的目标图像

**DirImageAfterMorphologyEx** = cvCloneImage(Timage3);

### 2.4.3 能量滤波的实现

在实现了高帽变换后，如图4.5，要从单一的二值化图中找到车牌区域，需要使用能量滤波，寻找区域能量符合要求的连通域。因此，能量滤波的效果直接影响到了车牌寻找的准确率。本系统中，首先将图像分成64块，分别统计每一小块的能量值，最后统计符合阈值要求的连通域。在对阈值的选取上，使用自适应阈值的方法。其关键代码及注释如下所示。

///计算能量

double Energy[8][8];

int height = DirImageAfterMorphologyEx\_Dst->height/8;

int width = DirImageAfterMorphologyEx\_Dst->width/8;

double SUM = 0.0; //记录总的能量值

double sum = 0.0; //记录每一块的能量值

//计算每一快的能量并存在energy数组之中

for(int x=0;x<8;x++)

{

for(int y=0;y<8;y++)

{

sum =Compute(); //计算每一个块的能量值

Energy[x][y]=sum;

sum=0;

SUM+=sum；

}

}

//找到合适的阈值，然后筛选区域

double E=(SUM/64.0); //阈值自减每一次减小的量

double Th\_max = 5\*E; //设定的最大阈值

double Th\_min = 0.5\*E; //设定的最小阈值

double Th=Th\_max; //采用自适应阈值的方式，初始阈值为最大值

bool flag = false; //标记是否找到合适的车牌位置

//自适应阈值寻找车牌

while(!flag&&Th\_max>Th\_min)

{

bool Energy\_flag[8][8]; //记录该区域是否符合阈值

for(int x=1;x<7;x++)

//合适的在flag图像中表示出来，不必计算边缘，在边缘的地方很可能出现照//不全车牌的情况，自动去掉

for(int y=1;y<7;y++)

{

if(Energy[x][y]>=Th)

{

Energy\_flag[x][y]=true;

for(int j=x\*height;j<((x\*height)+height);j++)

for(int i=y\*width+1;i<((y\*width)+width-1);i++)

Paintcolor(i,j); //将符合条件的区域涂白

}

}

//判断寻找到的连通域的长度是否大于两块

if(FindWidth > =2)

Output(); //符合要求，则输出

else

Th=Th-0.5\*E; //不符合要求，阈值减小，再次运行直到找到

}

### 4.4.4 Opencv内部的图像结构与Qt内部图像结构的转换

由于，图像要在Qt窗口中显示，由于OpenCV与Qt图像存储的机制不同。因此，我们需要将图像转换成Qt内部封装好的图像结构，然后再窗口中显示。系统运行的关键代码与注释如下所示。

//将文件显示在窗口中,转换成Qt显示图片的方式(BGR->RGB)

IplImage \*Timage2 = cvCreateImage(cvGetSize(Timage1),IPL\_DEPTH\_8U,3);

//将OpenCV图像的R分量与B分量的值互换

cvConvertImage(Timage1,Timage2, CV\_CVTIMG\_SWAP\_RB);

//将IplImage类型的图像转换成Qt封装的图像格式

QImage image = NULL; image((constuchar\*)Timage2->imageData,Timage2->width,Timage2->height,QImage::Format\_RGB888);

QPixmap pixmap1 = QPixmap::fromImage(image);

//显示图像

ui->Show->setPixmap(pixmap1);