武汉大学计算机学院

实验报告

课程名称 高级语言程序设计实验

专业年级 **2015** 级计算机科学与技术（卓越工程师）

姓　　名 李文洲

学　　号 **2015301500162**

协 作 者 无

实验学期 **2016-2017** 学年 第二 学期

填写时间 **2017** 年 **5** 月 **28** 日

目　录

1. 选题介绍

1.1 选题原因

本学期开展了《数字图像处理》课程与《高级语言程序设计》课程，因为对图像视频处理比较感兴趣，所以确定了如下选题——基于监控视频的运动物体轮廓检测与匹配。

1.2 选题内容

选题内容是基于监控视频的物体轮廓检测与匹配，提前预置 3 个样本.out 文件供程序匹配及 3 个.jpg 文件供程序显示匹配成功，程序运行时提示用户选择要进行匹配的视频文件，用户确定后将选择的视频和预置的 3 个视频样本匹配，若找到匹配则提示匹配成功并显示预置的匹配图片，否则提示匹配失败。

1.3 选题意义

运动目标检测是目前机器视觉领域的研究热点,广泛应用于安防保卫、机器人避障、人脸识别等领域。对于人体生物特征的识别，可以说从古到今一直受到人类的关注。随着智能监控、人机交互技术的发展，具有视频分析处理能力并可对运动目标实现检测和跟踪的智能化视频监控系统已成为研究的热点和主流。

2. 需求分析

2.1 需求介绍

该软件需要能够让用户选择需要匹配的源视频，让用户选择开始进行分析处理，并输出分析的结果，告知用户是否找到相似轮廓，若找到相似轮廓则输出匹配成功提示。

2.2 可行性分析

视频图像中运动目标检测相对于静态图像而言稍显复杂一些，运动目标检测是指在视频图像序列中判断是否有前景目标的运动，如果有前景目标，则对目标进行初始定位的检测。视频是由时间上连续的图像序列构成的，故对于视频中运动目标的检测是按照一定的周期从视频序列中提取出一张张静态图像帧来实现检测的，因此视频序列图像与静态图像中的目标检测方法存在相似的地方，而不同的地方就在于运动目标时间上的连惯性。针对视频图像的特殊性，我们常用于运动目标检测的方法有以下几种。

1. 帧间差分法

帧差法是最为常用的运动目标检测和分割方法之一，基本原理就是在图像序列相邻两帧或三帧间采用基于像素的时间差分通过闭值化来提取出图像中的运动区域。首先，将相邻帧图像对应像素值相减得到差分图像，然后对差分图像二值化，在环境亮度变化不大的情况下，如果对应像素值变化小于事先确定的阂值时，可以认为此处为背景像素:如果图像区域的像素值变化很大，可以认为这是由于图像中运动物体引起的，将这些区域标记为前景像素，利用标记的像素区域可以确定运动目标在图像中的位置。由于相邻两帧间的时间间隔非常短，用前一帧图像作为当前帧的背景模型具有较好的实时性，其背景不积累，且更新速度快、算法简单、计算量小。算法的不足在于对环境噪声较为敏感，闽值的选择相当关键，选择过低不足以抑制图像中的噪声，过高则忽略了图像中有用的变化。对于比较大的、颜色一致的运动目标，有可能在目标内部产生空洞，无法完整地提取运动目标。

2. 光流法

光流法的主要任务就是计算光流场，即在适当的平滑性约束条件下，根据图像序列的时空梯度估算运动场，通过分析运动场的变化对运动目标和场景进行检测与分割。通常有基于

全局光流场和特征点光流场两种方法。最经典的全局光流场计算方法是 L-K(Lueas&Kanada)法和 H-S(Hom&Schunck)法，得到全局光流场后通过比较运动目标与背景之间的运动差异对运动目标进行光流分割，缺点是计算量大。特征点光流法通过特征匹配求特征点处的流速，具有计算量小、快速灵活的特点，但稀疏的光流场很难精确地提取运动目标的形状。总的来说，光流法不需要预先知道场景的任何信息，就能够检测到运动对象，可处理背景运动的情况，但噪声、多光源、阴影和遮挡等因素会对光流场分布的计算结果造成严重影响;而且光流法计算复杂，很难实现实时处理。

3. 背景减除法

背景减除法是一种有效的运动对象检测算法，基本思想是利用背景的参数模型来近似背景图像的像素值，将当前帧与背景图像进行差分比较实现对运动区域的检测，其中区别较大的像素区域被认为是运动区域，而区别较小的像素区域被认为是背景区域。背景减除法必须要有背景图像，并且背景图像必须是随着光照或外部环境的变化而实时更新的，因此背景减除法的关键是背景建模及其更新。针对如何建立对于不同场景的动态变化均具有自适应性的背景模型，减少动态场景变化对运动分割的影响，研究人员已提出了许多背景建模算法，但总的来讲可以概括为非回归递推和回归递推两类。非回归背景建模算法是动态的利用从某一时刻开始到当前一段时间内存储的新近观测数据作为样本来进行背景建模。非回归背景建模方法有最简单的帧间差分、中值滤波方法、Toyama 等利用缓存的样本像素来估计背景模型。

最后，我决定使用混合高斯模型的背景减除法对视频进行运动目标的提取。

在提取出运动目标后，后续的步骤就较为明朗：先用腐蚀降噪算法对整体进行降噪去除噪点，再用 Canny 算子对目标进行边缘检测，提取出目标边缘，最后用 cvFindContours 算法提取目标轮廓并进行相关计算及过滤。

而对于用户交互的构建可以通过 Qt 框架进行实现。

2.3 小结

从需求分析和可行性分析来看，程序主要分为两大部分，一个部分是 UI 交互设计，一个部分是核心匹配功能。

3. 界面分析

3.1 界面原型设计

界面使用了 Qt 进行设计。

最顶上是这个程序的说明；往下是选取路径显示；再往下的两个 button 分别是选择视频源文件、确定开始分析；再往下的一行提示是否找到匹配；再往下的方框显示匹配图像（若有）。

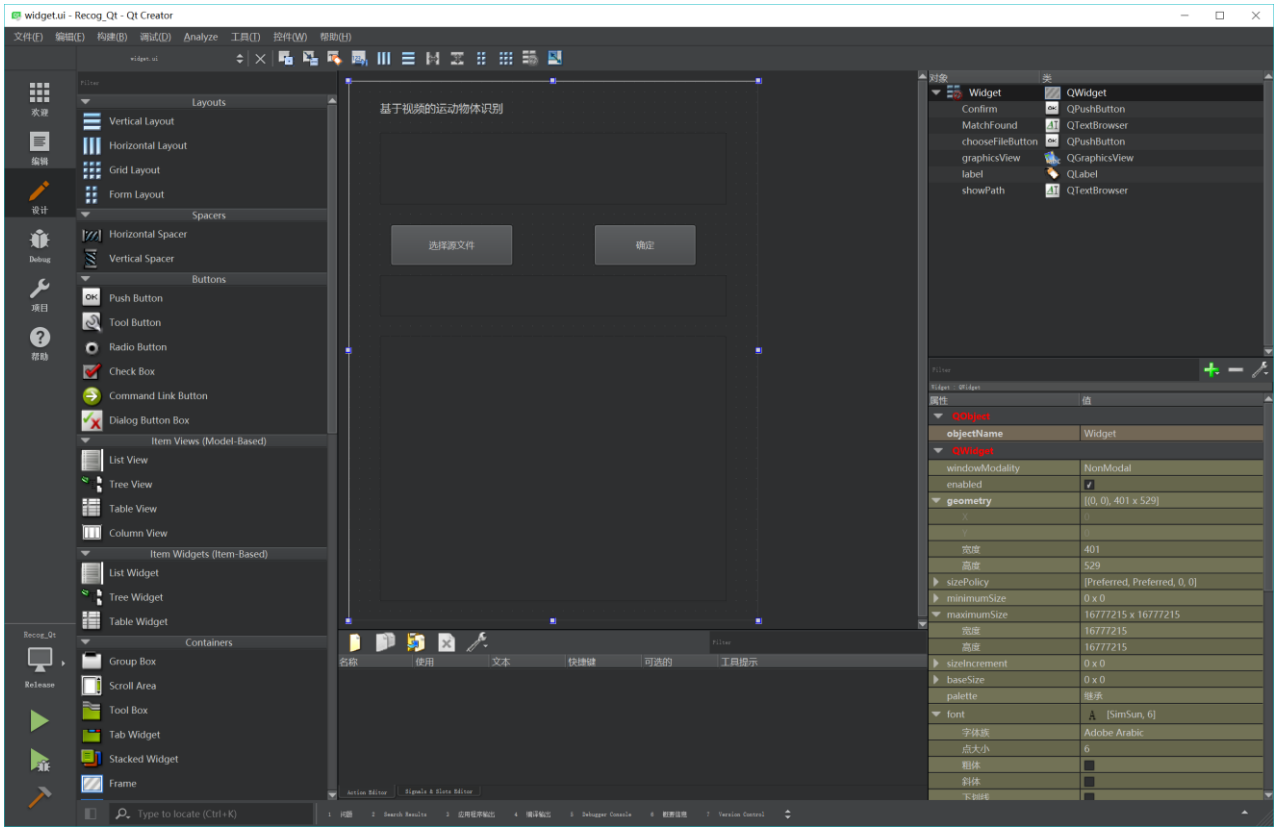

图 1-界面设计

初始显示如下：

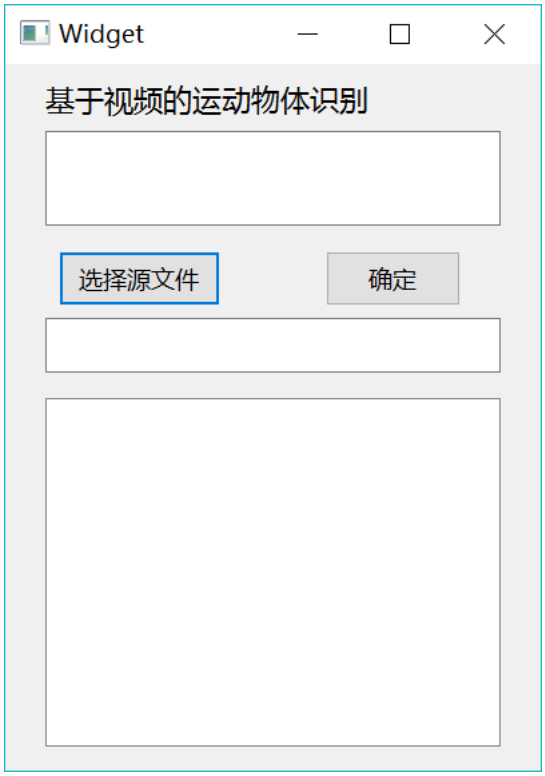

图 2：初始界面

成功匹配时显示如下：

图 3：成功匹配界面

未找到匹配时显示如下：

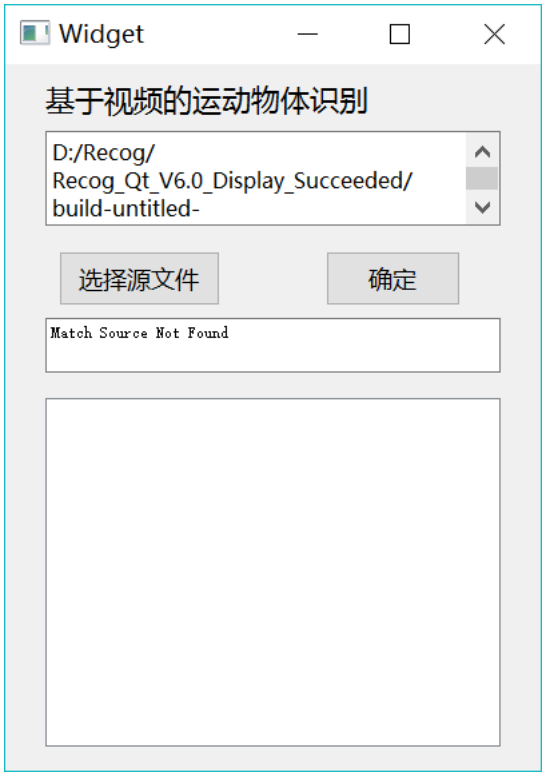

图 4：匹配失败界面

3.2 交互设计

程序使用 Qt 进行交互设计，总共使用了两个触发槽函数：chooseFileButton()、on_Confirm_clicked()，分别对应“选择源文件”、“确定”两个按钮触发。

```
public slots:
    void chooseFileButton();
    void on_Confirm_clicked();
```

图 5：槽设计

初始化时程序如图所示：

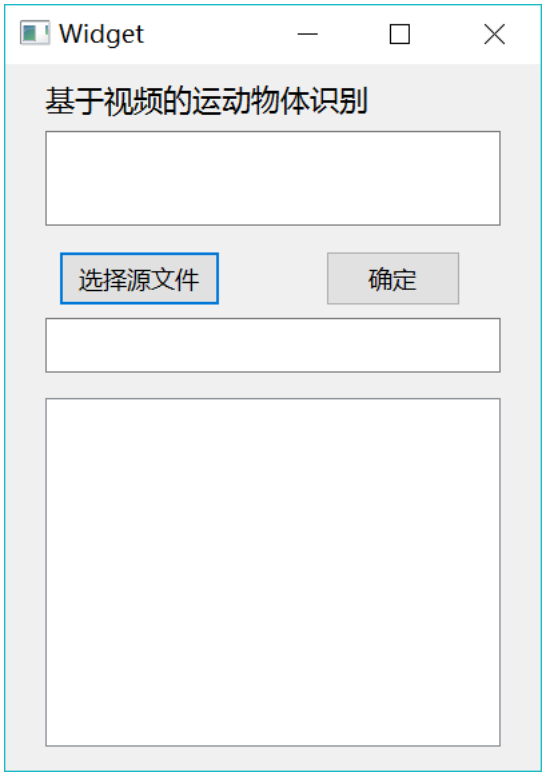

图 6：初始化设计

需要用户手动选择要匹配的视频文件，用户单击“选择源文件”按钮后触发 QFileDialog 事件：getOpenFileName，打开文件输入框，需要用户选择输入的文件。

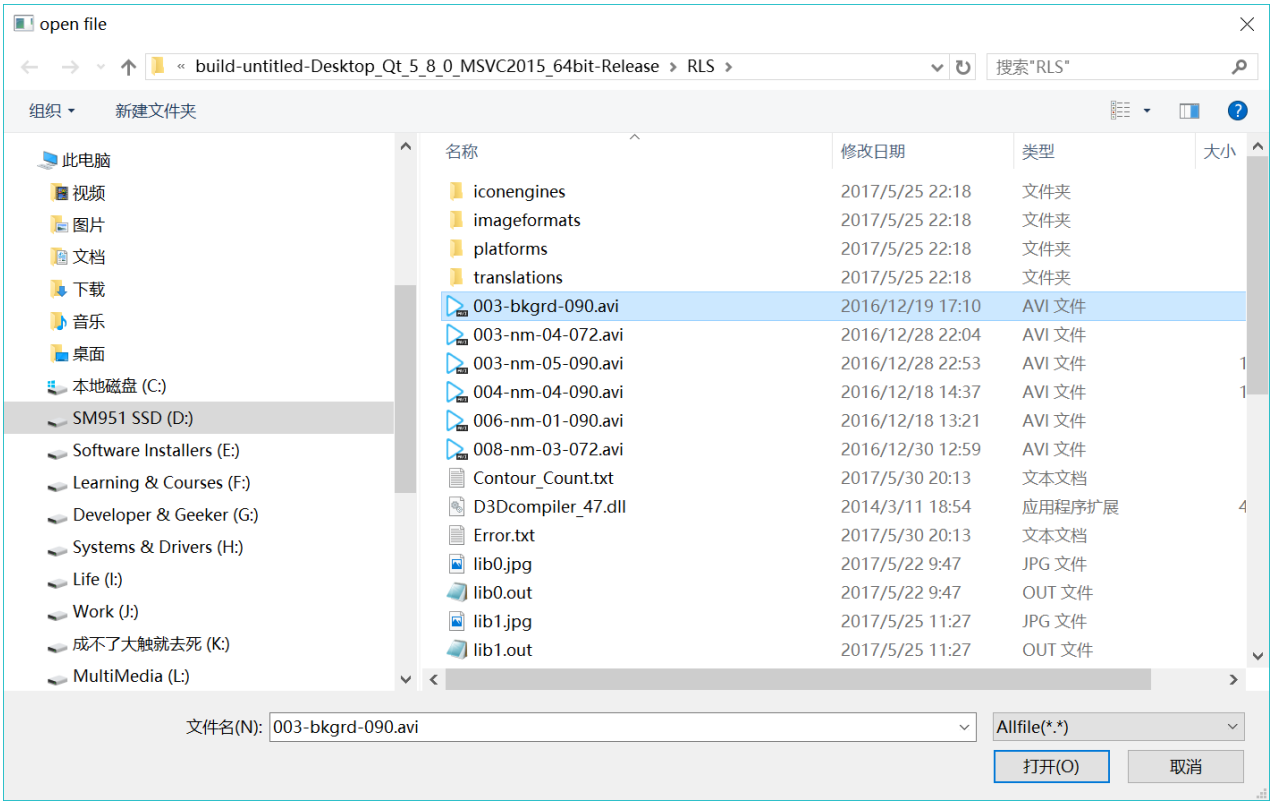

图 7：用户选择文件

在用户选择文件，单击“打开”后，文件的路径将会完整显示在上方 QTextBroswer 内容框 showPath 中。

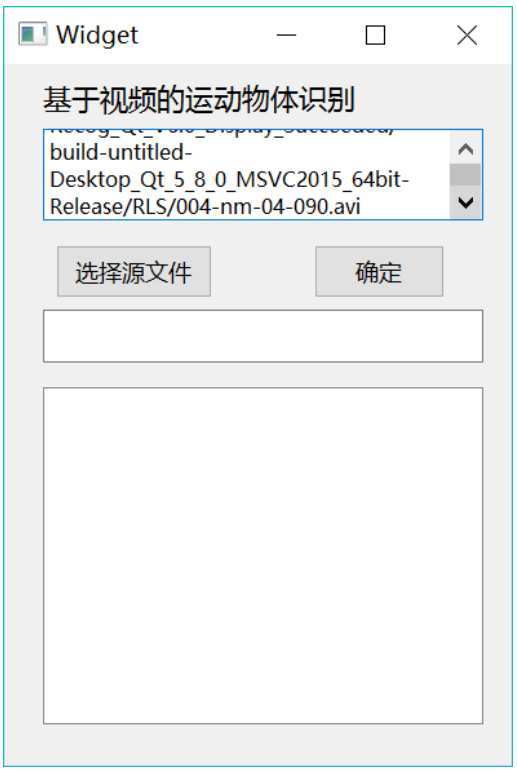

图 8：选择文件成功

源代码如下：

```
void Widget::chooseFileButton()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("open file"), " ",  tr("Allfile(*.*)"));
    //测试是否成功打开
    ui->showPath->setText(fileName);
    this->SourceFilePath = fileName;
}
```

图 9：文件选择源代码

在用户点击确定后，激活事件 on_Confirm_clicked()，开始运行匹配程序主体，程序会显示处理过程中产生的临时图像：

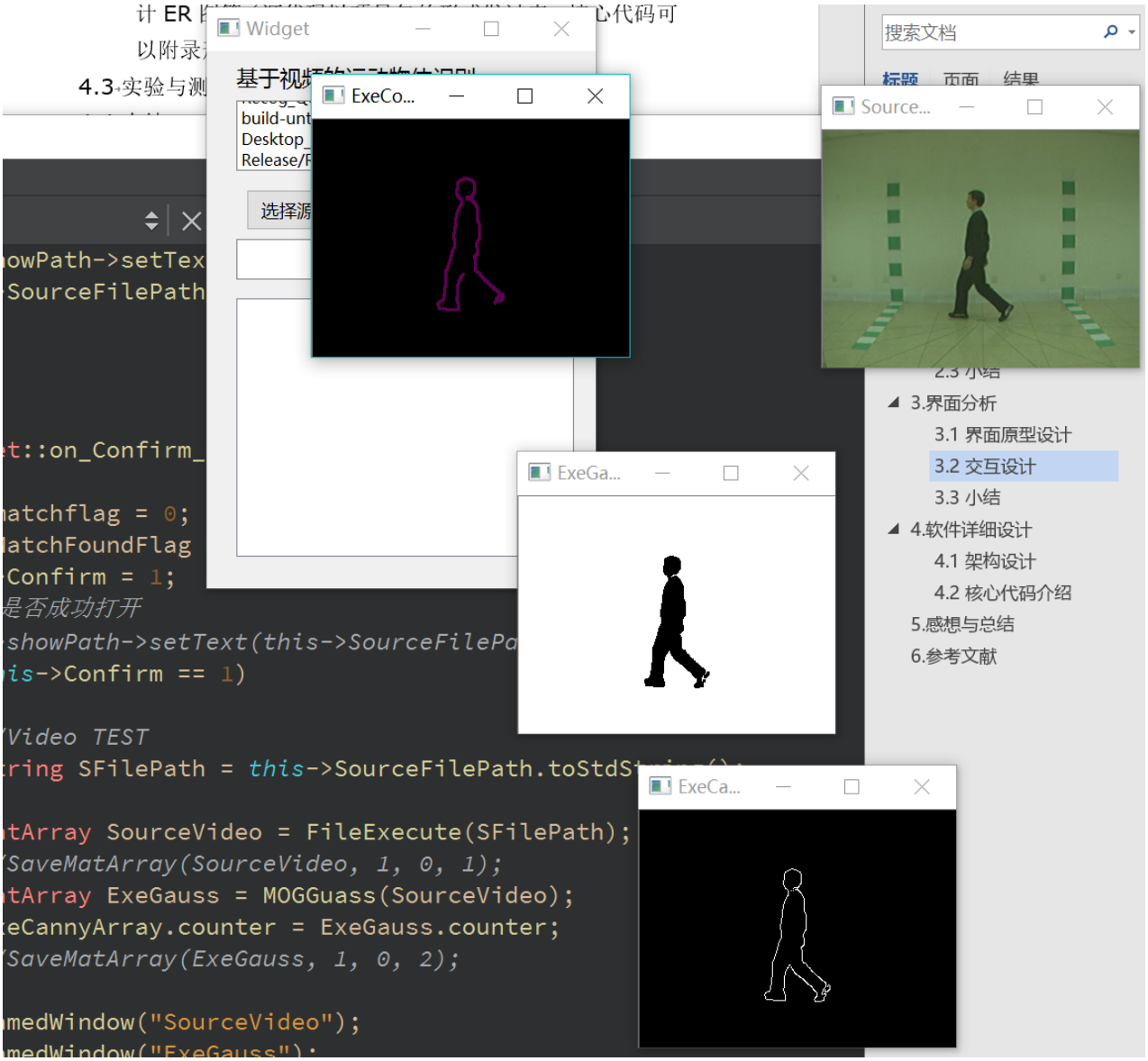

图 10：处理运行时界面

并在结束处理后关闭这些窗口，显示是否和预置轮廓匹配。

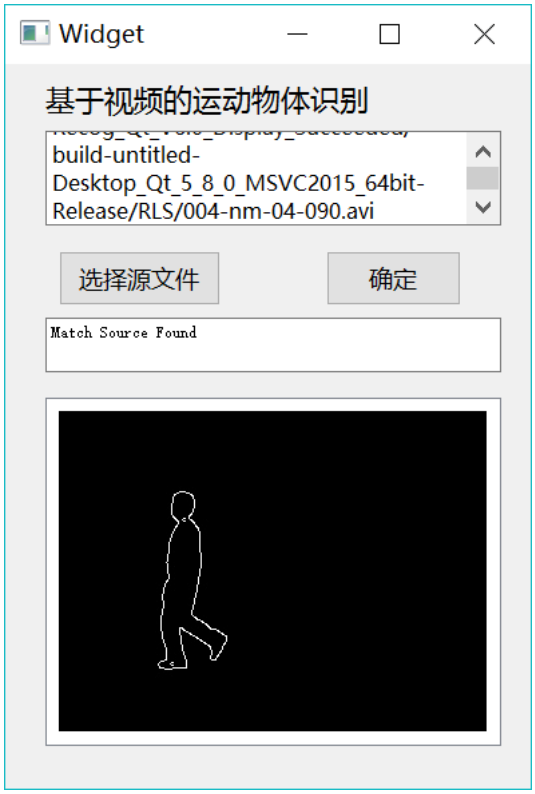

图 11：处理结束，成功匹配

3.3 小结

总的来说，本程序使用了 Qt 进行构建，较为方便地构建了界面及交互，点按事件选择了 QPushButton 类，内容显示选择了 QTextBrowswer 类，图像显示选择了 QGraphicsView 类，标签显示选择了 QLabel 类，并使用了 clicked()方法触发事件，构建起了整个程序界面。

4. 软件详细设计

4.1 架构设计

本项目实验环境：

System：Windows10 1607

IDE：Qt Community 4.7 + MicroSoft Visual Studio 2015

Lib：OpenCV 3.2

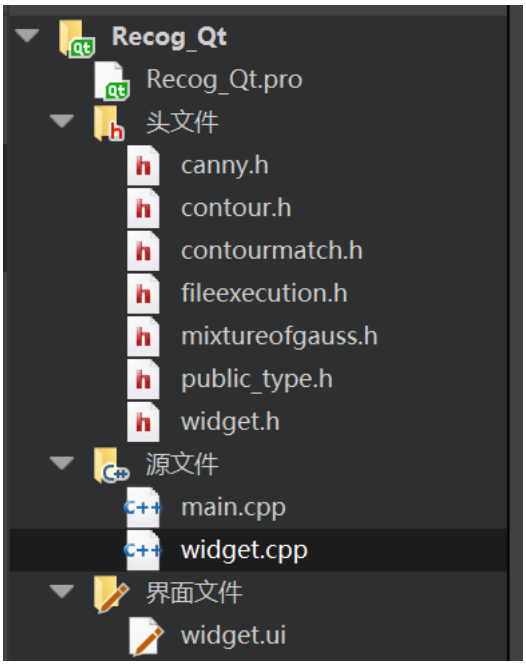

图 12：Qt 架构

本软件使用 widget.ui 文件做 UI 交互设计；用 widget.h 定义用到的公有、私有方法及槽；main.cpp 调用图形界面；widget.cpp 作为核心处理各种事件，并调用包含各种处理方法的头文件：边缘检测 canny.h、轮廓提取 contour.h、轮廓匹配 contourmatch.h、文件处理 fileexecution.h、混合高斯背景减除提取运动物体 mixtureofgauss.h、openCV 引用库定义及图像序列类型定义 public_typeh.h。

在 widget.cpp 中首先在 Widget 中定义了界面交互信号槽；再在 chooseFileButton()定义点击“选择源文件”后的事件；在 on_Confirm_clicked()定义点击“确定”后的事件；在 Display1 定义显示图片事件。在 on_Confirm_clicked()中，实现了读取指定路径的视频文件，并将帧缓存入 SourceVideo 这个图像序列中，并依次调用背景减除法 MOGGauss()、边缘检测

getCanny()、轮廓检测处理 getContour()、轮廓匹配 contourMatch()、显示图像 Display1()进行处理。

4.2 核心代码介绍

canny.h

```
Mat getCanny(Mat toExeCanny)
{   //using Canny to get the edge of object by Lwz 2017.05.01
    //in: Mat need to be Execute
    //out:Canny of input
    Mat CannyExecute;

    //Reduce Noise by DILATE
    //Set Struct Element
    Mat elementDILATE = getStructuringElement(MORPH_RECT, Size(3, 3));
    dilate(toExeCanny, toExeCanny, elementDILATE);

    //Reduce Noise by ERODE
    //Set Struct Element
    Mat elementERODE = getStructuringElement(MORPH_RECT, Size(3, 3));
    erode(toExeCanny, toExeCanny, elementERODE);

    Canny(toExeCanny, CannyExecute, 190, 300);
    threshold(CannyExecute, CannyExecute, 250, 255, THRESH_BINARY);
    return CannyExecute;
}
```

contour.h

```
Mat getContour(Mat src, int &contourCount, double minarea)
{   //using findContour to get the contour of object & count the contour by Lwz 2017.05.01
    //in: Mat need to be execute, container of contour counter, input the least area
    //out: Image of contour
    int counter = 0;
    //define output
    Mat dst, canny_output;
    if (!src.data)
    {   //open failed
        FILE *fp;
        fp = fopen("Error.txt", "a+");
        fprintf(fp, "\n getContour Open Image Error");
```

```
    fclose(fp);
}
else
{
    FILE *fp;
    fp = fopen("Error.txt", "a+");
    fprintf(fp, "\n getContour Open Image Succeeded");
    fclose(fp);
}
//pram. for findContours
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;

// Detect edges using canny
Canny(src, canny_output, 100, 300);
threshold(canny_output, canny_output, 10, 255, THRESH_BINARY);

// Find contours
findContours(canny_output, contours, hierarchy, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE);
//CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE

//mark out the max area contour
double maxarea = 0;
int maxAreaIdx = 0;

for (int i = 0; i < contours.size(); i++)
{
    double tmparea = fabs(contourArea(contours[i]));
    if (tmparea>maxarea)
    {
        maxarea = tmparea;
        maxAreaIdx = i;
    }
    //delete contours whose area is smaller than preset
    if (tmparea < minarea)
    {
        contours.erase(contours.begin() + i);
        //std::wcout << "delete a small area" << std::endl;
        continue;
    }
    else
        counter++;
```

```
    }

    // Draw contours with different colors
    dst = Mat::zeros(canny_output.size(), CV_8UC3);

    for (int i = 0; i< contours.size(); i++)
    {
        //Random Color
        RNG rng;
        Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255), rng.uniform(0, 255));
        drawContours(dst, contours, i, color, 2, 8, hierarchy, 0, Point());
    }

    contourCount = counter;
    return dst;
}
```

Contourmatch.h

```
Mat contourMatch(Mat ExeCanny, bool &MatchFlag)
{   //mark out contour if match source video by Lwz 2017.05.01
    //in:Canny-Executed Mat, Match flag
    //out:the match canny(preset)
    //size of video
    int const m = 320;
    int const n = 240;
    Mat FoundMatch;
    //Save ExeCanny To "temp.out"
    SaveMats(ExeCanny, 0, 1, 5, 0);

    //define tempo input name
    char *SampleName = new char[100];
    char *TempName = new char[100];
    char *MatchName = new char[100];
    for (int count = 0; count < 3; count++)
    {
        MatchFlag = 1;
        char SampleArray[m][n];
        char TempArray[m][n];

        //string IndexSampleNameString = std::to_string(imgIndex);
        //string FileName = FileSource + IndexSampleNameString;
```

```
            sprintf(SampleName, "lib%d.out", count);
            sprintf(TempName, "temp.out");
            FILE *fp_s;
            FILE *fp_t;
            //input .out to compare
            fp_s = fopen(SampleName, "a+");
            fp_t = fopen(TempName, "a+");
            if (fp_s == NULL || fp_t == NULL)
            {
                continue;
            }
            for (int i = 0; i < m; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    //input .out
                    fscanf(fp_s, "%c", &SampleArray[i][j]);
                    fscanf(fp_t, "%c", &TempArray[i][j]);
                    //compare
                    if (SampleArray[i][j] != TempArray[i][j])
                    {
                        MatchFlag = 0;
                        break;
                    }
                }
                fscanf(fp_s, "\n");
                fscanf(fp_t, "\n");
            }
            fclose(fp_s);
            fclose(fp_t);

            //if match mat found, trans the mat to FoundMatch and return to caller
            if (MatchFlag == 1)
            {
                sprintf(MatchName, "lib%d.jpg", count);
                FoundMatch = imread(MatchName);
                break;
            }
        }
        return FoundMatch;
}
```

fileexecution.h

```
/*****************************************************************
* File & VideoInput Execution by Lwz 2017.02.22
* Save Source Video Frames to a MatArray or a Mat
*****************************************************************/
//Open Video Source and Save Frames to an MatArray
MatArray FileExecute(string FileAddress)
{
    //Save VideoFrame
    MatArray VideoFrame;
    //Set FrameCounter
    VideoFrame.counter = 0;
    //open source video file
    VideoCapture sourceVideo(FileAddress);
    if (!sourceVideo.isOpened())
    {
        FILE *fp;
        fp = fopen("Error.txt", "a+");
        fprintf(fp, "\nVideo Open Error");
        fclose(fp);
        exit(1);
    }
    else
    {
        FILE *fp;
        fp = fopen("Error.txt", "a+");
        fprintf(fp, "\nVideo Open Succeed");
        fclose(fp);
    }
    //create frame false flag
    bool stop(false);
    while (!stop)
    {
        //execute only if frame exist
        if (!sourceVideo.read(VideoFrame.Frame[VideoFrame.counter]))
        {
            break;
        }
        VideoFrame.counter++;
    }
    VideoFrame.counter--;
    return VideoFrame;
}
```

```
//Save MatArray to disk file
bool SaveMatArray(MatArray ToSave, bool isSaveImg, bool isSaveOut, int SaveType)
{
    string MatType;
    switch (SaveType) {
    case 1:
    {
        MatType = "Source";
        break;
    }
    case 2:
    {
        MatType = "Gauss";
        break;
    }
    case 3:
    {
        MatType = "Contour";
        break;
    }
    case 4:
    {
        MatType = "Canny";
        break;
    }
    default:
        break;
    }

    char *imageSaveName = new char[100];
    char *matrixSaveName = new char[100];
    int imgIndex(0);

    for (int count = 0; count <= ToSave.counter; count++)
    {
        string imgIndexString = std::to_string(imgIndex);
        string FileName = MatType + imgIndexString;
        if (isSaveImg)
        {//Save mat as .jpg
            sprintf(imageSaveName, "image_%s.jpg", FileName.c_str());
            imwrite(imageSaveName, ToSave.Frame[imgIndex]);
```

```
    }

    if (isSaveOut)
    {//Save mat as .out
        sprintf(matrixSaveName, "mat_%s.out", FileName.c_str());
        IplImage temp = ToSave.Frame[imgIndex];

        int m = temp.height;
        int n = temp.width;

        //distribute Mem
        int **p;
        p = new int *[m];
        for (int i = 0; i < m; i++)
        {
            p[i] = new int[n];
        }

        FILE *fp;
        fp = fopen(matrixSaveName, "w");

        uchar *ptr;
        for (int i = 0; i < m; i++)
        {
            ptr = (uchar*)temp.imageData + i*temp.widthStep;
            for (int j = 0; j < n; j++)
            {
                p[i][j] = (int) *(ptr + j);
                //if Grey pixel vary from 100 to 255, shows 0
                if (p[i][j] > 100)
                    fprintf(fp, "0");
                //else pixel vary from 0 to 100, shows 1
                else
                    fprintf(fp, "1");
            }
            fprintf(fp, "\n");
        }
        fclose(fp);
    }
    imgIndex++;
}
return 1;
```

```
}
//Save Mat To Disk File
bool SaveMats(Mat ToSave, bool isSaveImg, bool isSaveOut, int SaveType, int imgIndex)
{
    string MatType;
    switch (SaveType) {
    case 1:
    {
        MatType = "Source";
        break;
    }
    case 2:
    {
        MatType = "Gauss";
        break;
    }
    case 3:
    {
        MatType = "Contour";
        break;
    }
    case 4:
    {
        MatType = "Canny";
        break;
    }

    case 5:
    {
        MatType = "temp";
    }
    default:
        break;
    }

    char *imageSaveName = new char[100];
    char *matrixSaveName = new char[100];

    //for (int count = 0; count <= ToSave.counter; count++)
    //{
    string imgIndexString = std::to_string(imgIndex);
    string FileName = MatType + imgIndexString;
```

```
if (isSaveImg)
{//Save mat as .jpg
    sprintf(imageSaveName, "image_%s.jpg", FileName.c_str());
    imwrite(imageSaveName, ToSave);
}

if (isSaveOut)
{//Save mat as .out
    sprintf(matrixSaveName, "mat_%s.out", FileName.c_str());
    if (MatType == "temp")
        sprintf(matrixSaveName, "temp.out");
    IplImage temp = ToSave;

    int m = temp.height;
    int n = temp.width;

    //distribute Mem
    int **p;
    p = new int *[m];
    for (int i = 0; i < m; i++)
    {
        p[i] = new int[n];
    }

    FILE *fp;
    fp = fopen(matrixSaveName, "w");

    uchar *ptr;
    for (int i = 0; i < m; i++)
    {
        ptr = (uchar*)temp.imageData + i*temp.widthStep;
        for (int j = 0; j < n; j++)
        {
            p[i][j] = (int) *(ptr + j);
            //if Grey pixel vary from 100 to 255, shows 0
            if (p[i][j] > 100)
                fprintf(fp, "0");
            //else pixel vary from 0 to 100, shows 1
            else
                fprintf(fp, "1");
        }
        fprintf(fp, "\n");
```

```
        }
        fclose(fp);
    }
    //}
    return 1;
}
//Qt Open File And Return File Address
QImage Mat2QImage(Mat Source)
{
    //Trans Mat To QImage
    if (Source.data)
    {
        //Mat TO QImage
        cv::cvtColor(Source, Source, CV_BGR2RGB);
        QImage tempQimg = QImage((const unsigned char *)(Source.data), Source.cols, Source.rows,
QImage::Format_RGB888);
        return tempQimg;
    }
}
```

mixtureofgauss.h

```
MatArray MOGGuass(MatArray SourceVideoMat)
{   //Execute Mixture of Guass by Lwz 2017.03.22
    //input: MatArray need to be execute
    //output: MatArray after Mixture Of Gauss Background Subtractor execute
    int counter = 0;
    //Define Foreground MatArray
    MatArray Foreground;
    Foreground.counter = SourceVideoMat.counter;
    //create Background Sub Tractor Mixture of Guass
    Ptr<BackgroundSubtractorMOG2> mog = createBackgroundSubtractorMOG2();
    //execute MOG
    while (counter <= SourceVideoMat.counter)
    {
        /*set mog var*/
        //contour Size
        mog->setNMixtures(25);
        //Var Initial
        mog->setVarInit(15);
        //BackGround Ratio
        mog->setBackgroundRatio(1);
        mog->setComplexityReductionThreshold(1000);
```

```
            //Shadow Detect
            mog->setDetectShadows(1);
            //Shadow Value
            mog->setShadowValue(10);
            //Shadow Thres
            mog->setShadowThreshold(5);
            //Var Thres
            mog->setVarThreshold(90);
            //Far Noise Thres
            mog->setVarThresholdGen(800);
            //Apply All VAR
            mog->apply(SourceVideoMat.Frame[counter], Foreground.Frame[counter], 0.00000000000000001);

            //cover RGB to Grey
            normalize(Foreground.Frame[counter], Foreground.Frame[counter], 0, 255, NORM_MINMAX);
            threshold(Foreground.Frame[counter], Foreground.Frame[counter], 220, 255, THRESH_BINARY_INV);

            counter++;
        }
        return Foreground;
}
```

widget.cpp

```
#include "widget.h"
#include "ui_widget.h"
#include "contour.h"
#include "fileexecution.h"
#include "mixtureofgauss.h"
#include "canny.h"
#include "contourmatch.h"
Widget::Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Widget)
{
    ui->setupUi(this);
    //signal slot
    connect(ui->chooseFileButton, SIGNAL(clicked()), this, SLOT(chooseFileButton()));
    imageLabel = new QLabel(this);
}

Widget::~Widget()
{
```

```
    delete ui;
}

void Widget::chooseFileButton()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("open file"), " ", tr("Allfile(*.*)"));
    ui->showPath->setText(fileName);
    this->SourceFilePath = fileName;
}

void Widget::on_Confirm_clicked()
{
    bool matchflag = 0;
    bool MatchFoundFlag = 0;
    this->Confirm = 1;
    //test if open successfully
    //ui->showPath->setText(this->SourceFilePath);
    if (this->Confirm == 1)
    {
        //Video TEST
        string SFilePath = this->SourceFilePath.toStdString();
        //Execute Input File
        MatArray SourceVideo = FileExecute(SFilePath);
        //SaveMatArray(SourceVideo, 1, 0, 1);
        MatArray ExeGauss = MOGGuass(SourceVideo);
        ExeCannyArray.counter = ExeGauss.counter;
        //SaveMatArray(ExeGauss, 1, 0, 2);
        //Set Windows to show the procedure
        namedWindow("SourceVideo");
        namedWindow("ExeGauss");
        namedWindow("ExeCanny");
        namedWindow("ExeContour");

        for (int i = 1; i < ExeGauss.counter; i++)
        {
            if (!ExeGauss.Frame[i].empty())
            {
                //Execute canny algorithm
                ExeCannyArray.Frame[i] = getCanny(ExeGauss.Frame[i]);
                //SaveMats(ExeCannyArray.Frame[i], 1, 1, 4, i);
                //set a var to save number of contour
                int ContourCount;
```

```
                //Execute findContour algorithm
                Mat ExeContour = getContour(ExeGauss.Frame[i], ContourCount, 1000);
                //Save number of contour to Disk File
                FILE *ct;
                ct = fopen("Contour_Count.txt", "a+");
                fprintf(ct, "%d\n", ContourCount);
                fclose(ct);
                waitKey(5);
                //Show execute procedure
                imshow("SourceVideo", SourceVideo.Frame[i]);
                imshow("ExeGauss", ExeGauss.Frame[i]);
                imshow("ExeCanny", ExeCannyArray.Frame[i]);
                imshow("ExeContour", ExeContour);
                if (ContourCount >= 1)
                {
                    //if there exist contour
                    Mat MatchFound;
                    //SaveMats(ExeContour, 1, 0, 3, i);
                    //execute contourMatch to find whether there are contour that match the source
                    MatchFound = contourMatch(ExeCannyArray.Frame[i], matchflag);
                    if (matchflag)
                    {
                        //if match found, display it
                        MatchFoundFlag = 1;
                        imwrite("MatchFound.jpg", MatchFound);
                        this->toDisplayImage = MatchFound;
                        Display1(matchflag);
                    }
                }
            }
        }
        //destroy the windows which showed the procedure
        destroyWindow("SourceVideo");
        destroyWindow("ExeGauss");
        destroyWindow("ExeCanny");
        destroyWindow("ExeContour");
    }
    if (!MatchFoundFlag)
    {   //if Match not found
        Mat nullImgM = imread("null.jpg");
        QImage nullImgQ = Mat2QImage(nullImgM);
        QGraphicsScene *scene = new QGraphicsScene;
```

```
        ui->MatchFound->setText("Match Source Not Found");
        scene->addPixmap(QPixmap::fromImage(nullImgQ));
        ui->graphicsView->setScene(scene);
        ui->graphicsView->show();
        update();
    }
}

void Widget::Display1(bool matchflag)
{
    if (matchflag)
    {   //if Match found, display it
        Mat Mimage = this->toDisplayImage;
        imwrite("MatchFound.jpg", Mimage);
        QImage img = Mat2QImage(Mimage);
        ui->MatchFound->setText("Match Source Found");
        ui->graphicsView->updatesEnabled();
        ui->graphicsView->update();
        ui->graphicsView->setViewportUpdateMode(QGraphicsView::FullViewportUpdate);

        //ui->graphicsView->resize(img.width(), img.height());
        QGraphicsScene *scene = new QGraphicsScene;
        scene->addPixmap(QPixmap::fromImage(img));
        ui->graphicsView->setScene(scene);
        //ui->graphicsView->adjustSize();
        ui->graphicsView->show();
        update();
    }
```

5. 感想与总结

因为时间原因，最终程序实现了预想中的大部分功能，但是仍有部分地方可进行进一步修改，诸如匹配的时候可以用到更高级的 PMS 等方法，使之变得更为可靠，更实用化。

在实现关于图形图像处理的过程中，遇到了 openCV 的诸多问题，通过翻阅 openCV Tutorial，上 stackOverfllow、github 等等多个网站才寻求到各种各样的答案，在这不断探求的过程中也积累了非常非常多的东西。OpenCV 提供各种各样开源的算法，但是合理的参数调配及算法组合是需要耐心不断尝试才能得到较好结果的。

而对于 Qt 的 UI 设计，这也是第一次使用 Qt 构建工程，了解了槽、信号这些调用与传送机制，合理利用这些东西能极大提升效率。

6. 参考文献

[1]《Qt Creator 快速入门 第二版》霍亚飞 著

[2]《Qt5 编程入门》霍亚飞 程梁 著