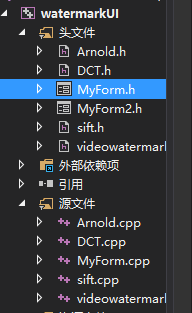
# 数字水印程序及软件使用说明

## 1、程序说明

### 1.1、源文件

 如右图所示，程序共包含6个头文件，5个cpp源文件。每个源文件对应一个头文件，另外MyForm2.h是对应一个弹出进度条的源代码。其中MyForm.h是主界面实现的代码，MyForm.cpp是程序的入口和界面交互的代码。Arnold、DCT和sift分别对应利用三种方法嵌入及提取水印的算法。Vediowatermark主要包含了视频的解码编码、镜头分割和SSIM的计算等。下面将每个文件单独说明。

### 1.2、 MyForm.h和MyForm.cpp

MyForm.h包括了主界面实现的类和对具体操作的交互函数的声明。包括下面几个函数：

Void outputbutton\_Click(System::Object^ sender, System::EventArgs^ e);

//让用户指定嵌入水印后的输出视频文件的保存路径

Void inputbutton\_Click(System::Object^ sender, System::EventArgs^ e);

//用户指定待嵌入水印的视频文件

Void arnoldstart\_Click(System::Object^ sender, System::EventArgs^ e);

//用Arnold算法开始嵌入水印

Void loadimage\_Click(System::Object^ sender, System::EventArgs^ e);

//加载要嵌入的水印图像

void embed\_start (VideoWatermark& processor);

//解码视频开始嵌入水印并编码

void detect\_start (VideoWatermark& processor);

//解码视频检测水印

Void detectfilebutton\_Click(System::Object^ sender, System::EventArgs^ e);

//用户指定要检测的视频文件

Void detect\_arnoldbutton\_Click(System::Object^ sender, System::EventArgs^ e);

//用Arnold算法开始检测水印

Void embeddedwatermark\_button\_Click(System::Object^sender, System::EventArgs^ e);

//加载嵌入视频的原始水印图像用于计算SSIM

MyForm.cpp文件包含了以上函数的详细实现。

### 1.3、vediowatermark.h和vediowatermark.cpp

Vediowatermark主要实现了视频的解码及编码。

bool readNextFrame(cv::Mat& frame)

//解码器读取视频下一帧

void writeNextFrame(cv::Mat& frame)

//编码器写入视频下一帧

bool setInput(std::string filename)

//设置输入视频文件

bool setOutput(const std::string &filename)

//设置输出视频文件

void setFrameEmbed(void (\*frameEmbedCallback)(cv::Mat&, const cv::Mat&,double&))

//设置嵌入水印用的方法

void setFrameDetect(void(\*frameDetectCallback)(cv::Mat&, cv::Mat&, double&))

//设置检测水印用的方法

void displayInput(std::string wn)

//播放输入视频

void displayOutput(std::string wn)

//播放嵌入后的视频

cv::Size getFrameSize()

//得到视频的帧长宽

double getFrameRate()

//得到视频的帧率

long getTotalFrameCount()

//得到总视频帧数

int getCodec(char codec[4])

//得到视频的编码格式

bool isStopped()

//让程序终止

以上函数因为比较短并且需要反复使用所以其实现直接在类的定义里面，成为内联函数。下面的函数实现在vediowatermark.cpp里面。

void ffmpegEncoderInit();

//编码器初始化

void ffmpegEncoderEncode(FILE \*file, uint8\_t \*data, long fnumber);

//编码并写入数据到文件

void ffmpegEncoderClose();

//关闭编码器

void extractFinalWatermark(cv::Mat &watermark, std::vector<std::vector<int>> &watermark\_accumulation);

//提取水印图像

cv::Mat GetHistValue(const cv::Mat image);

//计算每帧图像的直方图用来做镜头分割

cv::Scalar getMSSIM(const cv::Mat& i1, const cv::Mat& i2);

//计算提取到水印和原始水印的结构相似度

### 1.4、Arnold.h和Arnold.cpp

Arnold.cpp主要实现了Arnold变换及其逆变换以及利用Arnold算法嵌入水印和提取水印，主要包含4个函数。

void Arnold(cv::Mat& src, int times);

//Arnold变换

void Reverse\_Arnold(cv::Mat& src, int times);

//Arnold逆变换

void ArnoldEmbedWatermark(cv::Mat& image, const cv::Mat& watermark, double &psnr);

//利用Arnold算法嵌入水印

void ArnoldDetectWatermark(cv::Mat& src, cv::Mat& detect\_watermark, double &nc);

//利用Arnold算法提取水印

### 1.5、sift.h和sift.cpp

Sift算法主要是把谭伟的MATLAB算法c++实现了，但是他的算法是用在图像水印的，对于视频水印无效，主要包含以下函数：

IplImage \* embedWatermarkInSector(IplImage \*I, int W[], int n, std::vector<cv::KeyPoint> keypoints, int m);

//对每个sift点的环域嵌入水印

void EmbedWatermark(cv::Mat& image, cv::Mat& output, long frame\_num);

//sift点检测并嵌入水印

bool detecWatermarkInSector(IplImage\* I, int x, int y, double sift\_orientation, double scale, int num\_watermark, char watermark[], char detected\_watermark[]);

//对每个sift点的环域检测水印

unsigned char\* DetecWatermark(cv::Mat &image, long frame\_num);

//sift检测并且提取水印

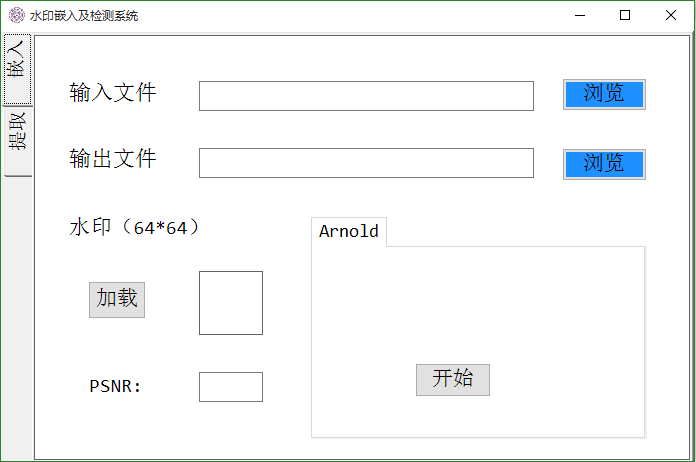
### 1.6、DCT.h和DCT.cpp

DCT方法目前只做了DCT变换和DCT反变换。

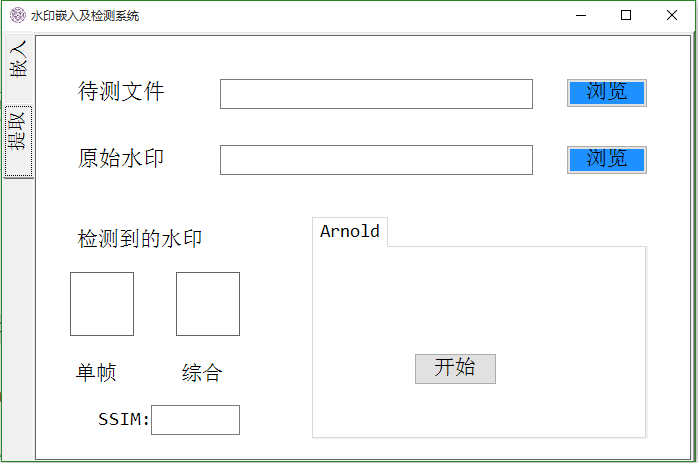
void DctEmbedWatermark(const cv::Mat& image, cv::Mat& output, long frame\_num)

## 2、软件的使用

运行程序后得到如下界面：下图是嵌入水印界面，我们可以选择输入文件设置输出文件，输出文件默认.h264格式。点击加载水印按钮可以选择需要嵌入的水印图像。



下图是提取水印的界面。用户可以选择待测文件，如果要求检测到水印和原始水印的结构相似度，可以选择原始水印。



在嵌入或提取水印时会弹出进度条如下所示：

