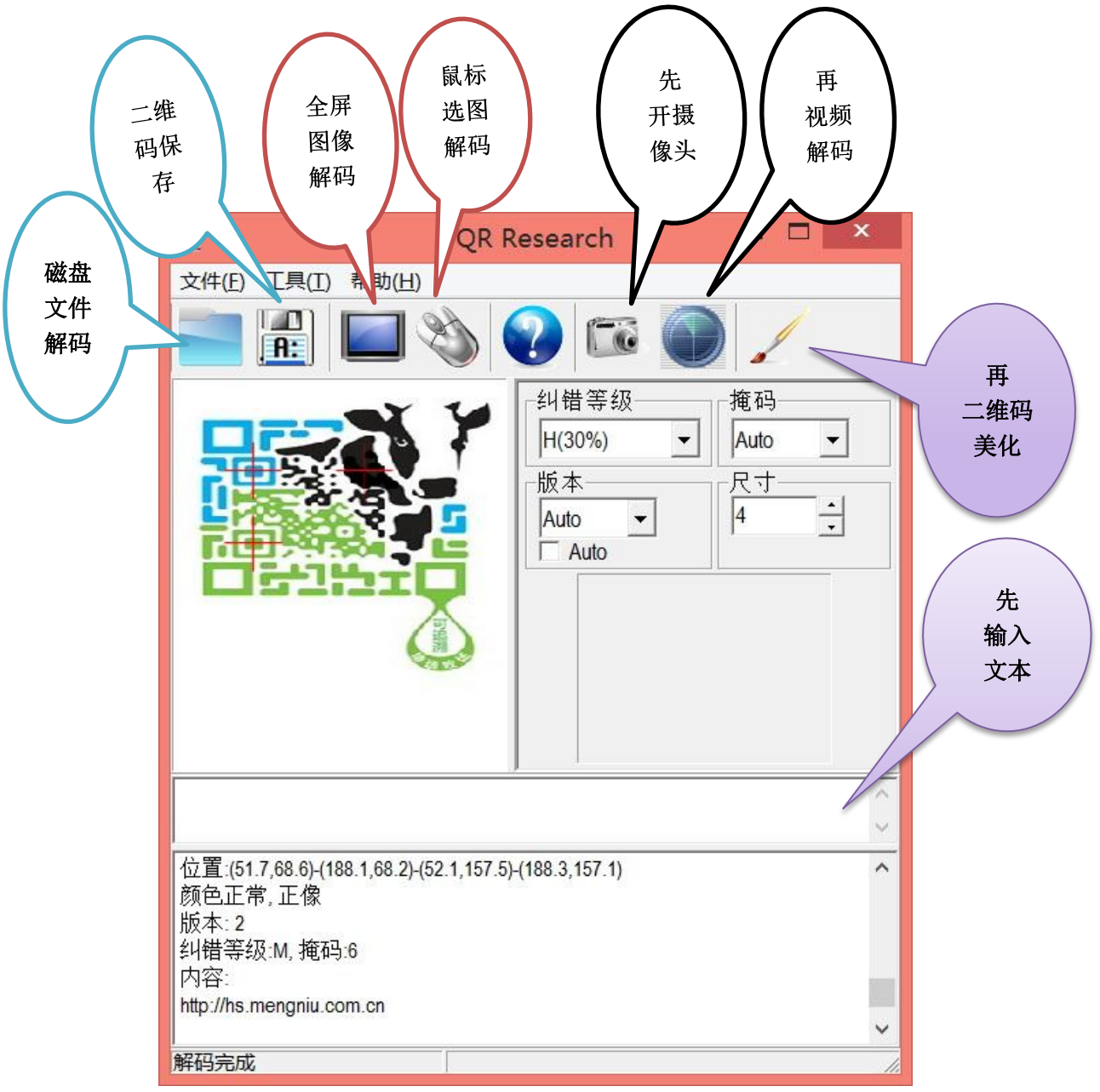
一. 功能说明

1. 软件界面



2. 解码功能如下：

- 同一幅图中最多解码 4 个二维码。
- 位置：正常二维码、镜像二维码（翻转放置）
- 颜色：正常色调二维码、反色的二维码。
- 缺角：二维码缺少任一个角

示例图像如下：

解码结果：

已解码数据 1:

位置:(476.8,209.0)-(531.9,307.5)-(323.0,295.1)-(378.0,393.5)

颜色正常，镜像

版本: 3

纠错等级:H，掩码:1

内容:

soochow university

已解码数据 2:

位置:(135.1,225.4)-(252.5,225.6)-(135.2,362.5)-(252.1,360.2)

颜色正常，正像

版本: 5

纠错等级:Q，掩码:1

内容:

http://pub.d777.jp/superman_campaign/?_prcode=10ad01nzm

已解码数据 3:

位置:(188.9,63.8)-(189.0,152.9)-(100.0,64.1)-(100.1,153.0)

颜色正常，镜像

版本: 2

纠错等级:M，掩码:6

内容:

http://hs.mengniu.com.cn

已解码数据 4:

位置:(318.7,22.7)-(515.5,22.9)-(318.7,212.0)-(515.4,212.4)

颜色反色，正像

版本: 3

纠错等级:Q，掩码:4

内容:

http://plinks.me/t9fxdk

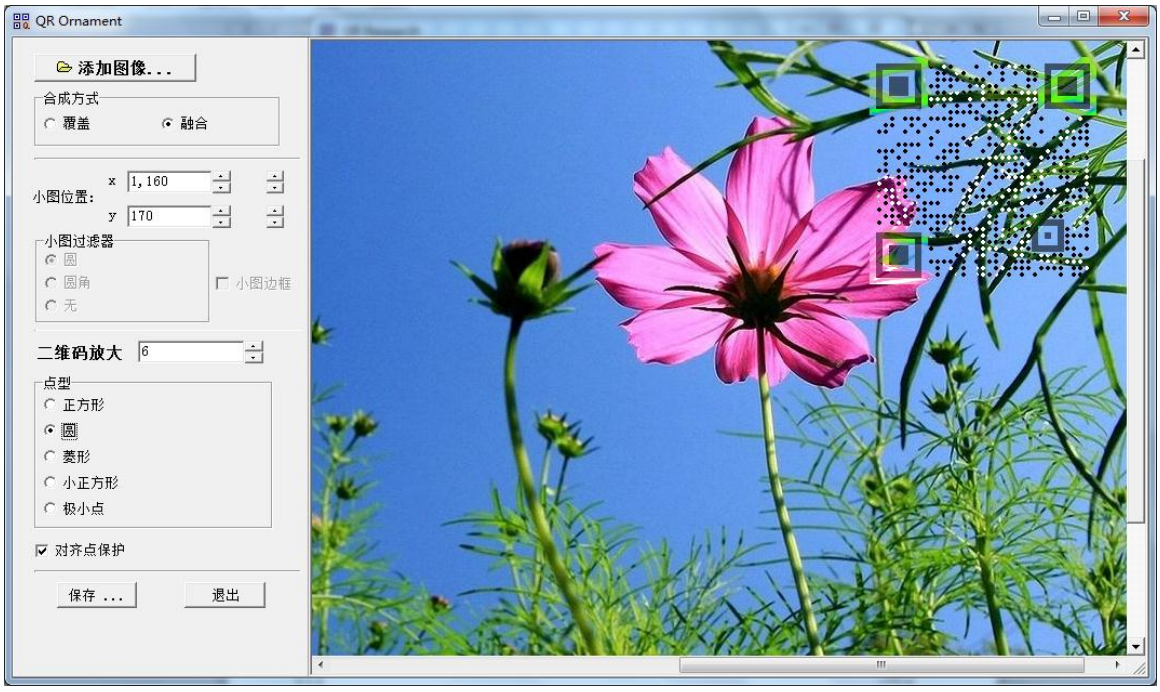
---

3. 编码

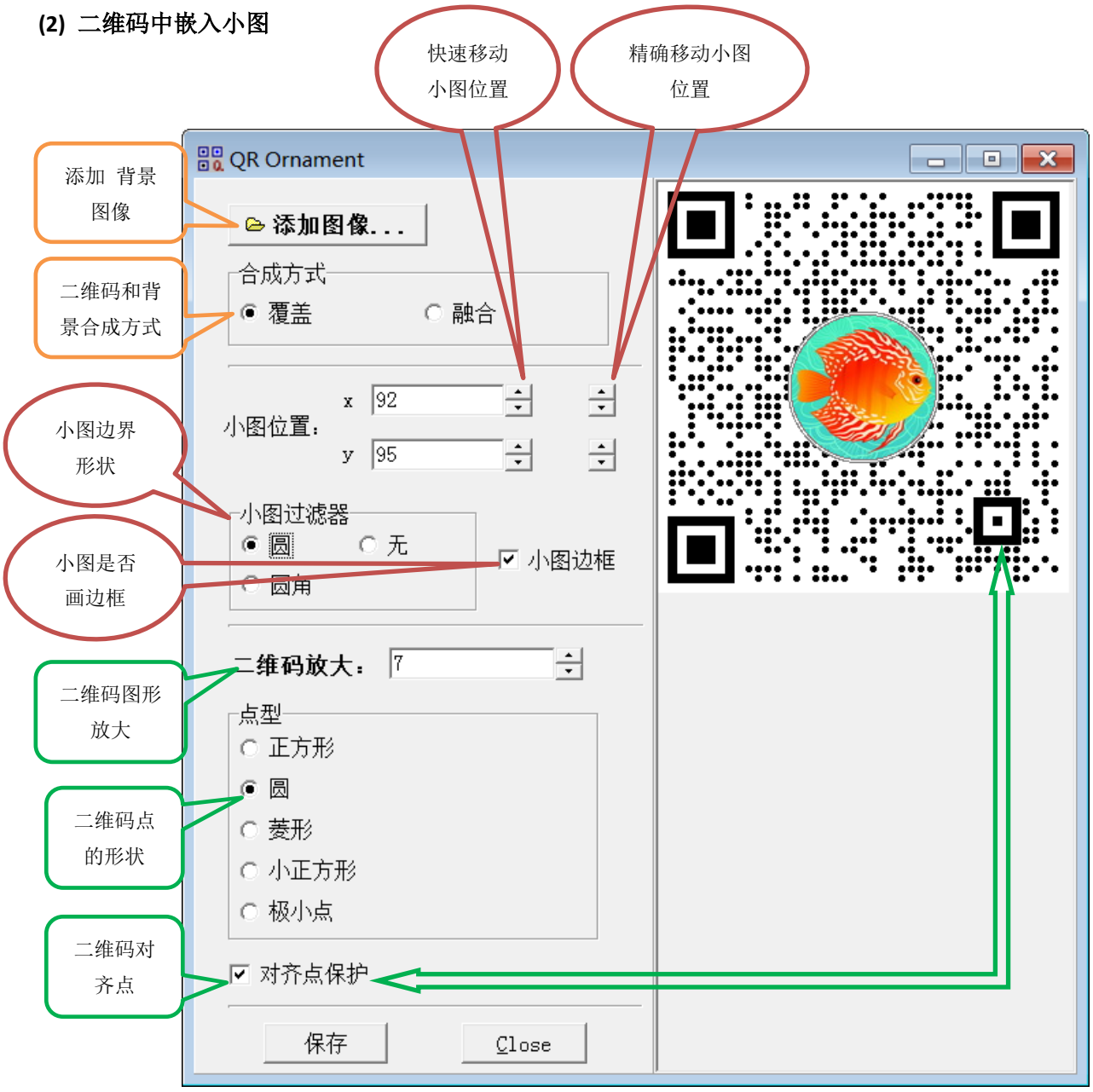
4. 美化

(1) 图像中嵌入二维码

采用图像融合算法， 形成**梦幻二维码**，效果如下：



(2) 二维码中嵌入小图



嵌入小图效果：

二．软件开发库说明

1．开发库形式

开发库 支持 VC、 VB、 C#、 DELPHI、BCB 开发语言；提供 VB、VC 的 Demo 程序。
开发库为 DLL + 加密狗 形式。

解码功能
（1）对 BMP,JPG,PNG 等多种格式文件进行解码
（2）对 8 位灰度的矩阵数据进行解码
（3）解码能力等同于软件 QR_Research_1.0

编码功能
（1）生成 BMP 图片
（2）生成像素矩阵，由用户自行进行图片生成

2．动态库声明

```
//-------------------------------------------------------------------------------------------
#define CH_MaxParseNum 4
#define MAX_SrcDataLen     7090

typedef struct _C_QrDecodedData
{
    int nParsedNums;                                // 解码的二维码数量

    int nInverseColor[CH_MaxParseNum];     // 0: 颜色正常；1:反色
    int nMirror[CH_MaxParseNum];              // 0: 正像       ；1:镜像

    int nVersion[CH_MaxParseNum];             // 版本号
    int nLevel[CH_MaxParseNum];               // 纠错等级
    int nMaskingNo[CH_MaxParseNum];          // 掩码

    int nParseStrLen[4];                           // 解码后，有效字符串长度
    char sParseStr[4][MAX_SrcDataLen];     // 解码后的字符串

    float dXY_FP[CH_MaxParseNum][2][4];   // 二维码寻像图形坐标，
    float dXY_4Corners[CH_MaxParseNum][2][4]; //二维码 4 个角的坐标

    int iReserved[16];       //保留

} C_QrDecodedData;

typedef struct _C_QrEncodedData
{
    unsigned char byEncoded_Data[177][177];     //编码后的数据矩阵
```

```
    int iEncoded_WH;        //有效的行和列：21-177
    int iReserved[16];     //保留
} C_QrEncodedData;

//-------------------------------------------------------------------------------------------
typedef int (CALLBACK* C_QR_Decode_Matrix)(int iBufType,unsigned char * src,int iImgWidth,int iImgHeight,int iQrMsg_Max,int iDecodeUTF8,C_QrDecodedData * p_QrDecodedData);
typedef int (CALLBACK* C_QR_Decode_File)(char * FilePath,int iQrMsg_Max,int iDecodeUTF8,C_QrDecodedData * p_QrDecodedData);

typedef int (CALLBACK* C_QR_Encode_Matrix)(int nLevel, int *nVersion, int bAutoExtent, int *nMaskingNo,char * lpsSource, int nSource,C_QrEncodedData * p_QrEncodedData);
typedef int (CALLBACK* C_QR_Encode_File)(int nLevel, int *nVersion, int bAutoExtent, int *nMaskingNo,char * lpsSource, int nSource,char * FilePath,int iMultiple,C_QrEncodedData * p_QrEncodedData);

typedef int (CALLBACK* C_QR_Set_DecodeTime_Limit)(int iTime_ms);
typedef int (CALLBACK* C_QR_Get_DecodeTime_Limit)(int *pTime_ms);
//-------------------------------------------------------------------------------------------
```

3. 动态库的编、解码函数说明：

（1）QR_Decode_File(char * FilePath,int iQrMsg_Max,int iDecodeUTF8,C_QrDecodedData * p_QrDecodedData);

```
 ------------------------------------------------QR_Decode_File-----------------------------------------------
/*功能：根据文件解码二维码

  参数：
  FilePath          ：  需要解码的图片文件名
  iQrMsg_Max        ：  限制最大解码数量，一张图最多解 4 个二维码；  1-4
  iDecodeUTF8       ：  0：直接解码；1：解码的结果，再使用 UTF8 解码
  p_QrDecodedData   ：  存放解码结果的数据缓存

  返回：
      >0    ：    成功
      -100  ：    没有找到加密狗
      其他  ：    解码失败
*/
```

（2）QR_Decode_Matrix(int iBufType,unsigned char * src,int iImgWidth,int iImgHeight,int iQrMsg_Max,int iDecodeUTF8,C_QrDecodedData * p_QrDecodedData);

```
-------------------------------------------------QR_Decode_Matrix-------------------------------------------------
/*功能：根据像素矩阵解码二维码

  参数：
  iBufType          ：  像素矩阵的类型,范围：0-1;
```

0:连续存放的二维矩阵;行与行是连续存放的
1: (unsigned char **)型的二维矩阵，即矩阵的行与行 可能不是连续存放的!

src : 像素数据
iImgWidth : 像素的宽度
iImgHeight : 像素的高度
iQrMsg_Max : 限制最大解码数量，一张图最多解 4 个二维码； 范围：1-4
iDecodeUTF8 : 0：直接解码；1：解码的结果，再使用 UTF8 解码
p_QrDecodedData : 存放解码结果的数据缓存

返回：
>0 : 成功
-100 : 没有找到加密狗
其他 : 解码失败
*/

（3）QR_Encode_File(int nLevel, int *nVersion, int bAutoExtent, int *nMaskingNo,char * lpsSource, int nSource,char * FilePath,int iMultiple,C_QrEncodedData * p_QrEncodedData);

----------QR_Encode_File----------
/* 功能：根据送入的参数，生成二维码文件

参数：
nLevel : 纠错级别 0 - 3; (0:7%; 1:15%; 2:25%; 3:30%)
nVersion : 版本号 0 - 40; 0 为自动
bAutoExtent : 0:版本由 nVersion 限定； 1:nVersion 不够容纳编码时，自动扩充版本号
nMaskingNo : 掩码，0:自动； 1-8 指定掩码号

lpsSource : 内容
ncSource : 内容长度， >0: 指定长度； <= 0：自动计算长度；

FilePath : 存放的图片文件名；Bmp 文件!
iMultiple : 模块像素（扩倍） 1-20

p_QrEncodedData : 存放编码结果的数据缓存

返回：
>0 : 成功
-100 : 没有找到加密狗
其他 : 编码失败

*/

（4）QR_Encode_Matrix(int nLevel, int *nVersion, int bAutoExtent, int *nMaskingNo,char * lpsSource, int nSource,C_QrEncodedData * p_QrEncodedData);

----------QR_Encode_Matrix----------

/*功能：生成二维码像素数据

参数：
nLevel : 纠错级别 0 - 3; (0:7%; 1:15%; 2:25%; 3:30%)
nVersion : 版本号 0 - 40; 0 为自动
bAutoExtent : 0:版本由 nVersion 限定； 1:nVersion 不够容纳编码时，自动扩充版本号
nMaskingNo : 掩码，0:自动； 1-8 指定掩码号

lpsSource : 内容
ncSource : 内容长度， >0: 指定长度； <= 0：自动计算长度；
p_QrEncodedData : 存放编码结果的数据缓存

返回：
>0 : 成功
-100 : 没有找到加密狗
其他 : 编码失败

*/

（5）QR_Set_DecodeTime_Limit(int iTime_ms);
----------QR_Set_DecodeTime_Limit----------
/*功能：设置解码允许的最大时间；防止图像太大，消耗很长时间

参数：
iTime_ms : 单位：毫秒；一般设置为 4000；（启动时，已经默认为 4000ms）

返回：
>0 : 成功
其他 : 失败
*/

（6）QR_Get_DecodeTime_Limit(int *pTime_ms);
----------QR_Get_DecodeTime_Limit----------
/*功能：获得解码允许的最大时间

参数：
pTime_ms : 数据缓存

返回：
>0 : 成功
其他 : 失败
*/