

Manual, Part IV

Modules



Thessaloniki 1997

Contents

1	Watermark module	1
1.1	Introduction	1
1.2	User's Guide	1
1.2.1	Installation	1
1.2.2	Casting a watermark	1
1.2.3	Detecting a watermarked image	3
1.3	Library functions	3
1.3.1	add_wtr_watermark	3
1.3.2	detect_wtr_watermark	4

Watermark module

1.1 Introduction

The Watermark module for EIKONA for Windows is an extension module containing an innovative algorithm for casting and detecting watermarks in digital images. Image watermarking is a technique for embedding a signal called *signature* or *watermark* to a digital image, so that unauthorized copying of the image is prevented. The watermark completely characterizes the person who applied it, and, therefore, proves the origin of the image. Section 1.3 of this manual describes the EIKONA image processing library functions for watermarks. Section 1.2 is a user's guide to the Watermarks module of EIKONA.

1.2 User's Guide

1.2.1 Installation

To install the Watermark module just copy the “`watermarks.dll`” file in the directory where EIKONA is installed. The module will be automatically loaded by EIKONA the next time you run it.

1.2.2 Casting a watermark

In order to apply a digital signature to an image, we use the option **Modules, Watermarking, Cast Watermark**. In the dialog box, we select the source and destination image buffers ; checking the *Color image* button if the image to be signed is a color one. field. In the *Key* field we provide a positive number between



Figure 1.2.1 Initial image “St.Gregorios”.

0 and $2^{32} - 1$ to be used as a seed for the encryption algorithm. Then, from the *Watermark level* group we select the desired embedding level . The embedding level specifies the robustness of the watermark. As the embedding level increases, the watermark becomes more robust under image modifications, such as JPEG compression and digital filtering. However, perceptually visible effects may arise.

Figure 1.2.1 shows the initial grayscale image “St.Gregorios”, and figure 1.2.2 shows a signed version of “St.Gregorios”, produced with encryption key set to 1000, and embedding level 2.



Figure 1.2.2 Signed image “St.Gregorios”.

1.2.3 *Detecting a watermarked image*

A signed image can be easily detected, if the encryption key is known. In order to detect a watermark from an image, we use the option **Modules, Watermarking, Detect Watermark**. In the corresponding dialog box, we provide the buffer where the signed image is stored, and the encryption key which was used to sign the image. We check the *Color Image* field if the source image is a color image. After pressing *OK* the program comes up with a message specifying whether the image was found signed with the given key or not.

1.3 Library functions

1.3.1 *add_wtr_watermark*

Subroutine to add a watermark on an image

LIBRARY MODE:

```
int add_wtr_watermark ( key, in1, in2, in3, ih, iv,
                        Cf, out1, out2, out3 )
```

KEY *key;

image in1, in2, in3, out1, out2, out3;

int ih, iv;

int Cf;

key : pointer to a structure KEY long seed, int wpow.

in1, in2, in3 : original RGB image buffers (in2=in3=NULL for BW case)

ih, iv : horizontal and vertical size of the original image

Cf : flag (0:BW image else Color image)

out1,out2,out3 : watermarked image RGB buffers (out2=out3=NULL for BWcase)

SHORT DESCRIPTION : This function determines the pixel positions for watermarking, creates a watermark mask and finally superimposes the watermark on the original image producing the watermarked image. The embedding level key (wpow) must be in the range [0,4]. The bigger it is, the bigger is the resistance (and the visibility of the watermark).

RETURN VALUES: 0, -1, -9, -21, -36, -400

SEE ALSO : detect_wtr_watermark().

1.3.2 *detect_wtr_watermark*

Subroutine to detect the existence of a watermark in an image.

LIBRARY MODE:

```
int detect_wtr_watermark ( key , in1 , in2, in3, ih , iv , Cf , p , R)
```

```
KEY   *key;
```

```
image in1, in2, in3, out1, out2, out3;
```

```
int    ih, iv;
```

```
int    Cf;
```

```
double *p;
```

```
float  *R;
```

```
key      : pointer to a structure KEY long seed, int wpow.
```

```
in1,in2,in3 : RGB image buffers (in2=in3=NULL for BW case)
```

```
ih, iv     : horizontal and vertical size of image
```

```
Cf         : flag (0:BW image else Color image)
```

```
p          : pointer to the calculated significance from t-test
```

```
R          : pointer to the ratio of the correctly detected pixels per total  
              watermarked pixels.
```

SHORT DESCRIPTION : This function searches for the watermark specified by the *key*. The statistical t-test is applied to determine the difference of intensity means of two image subsets determined by the watermark. T-test returns the significance P_{err} of the difference between the mean values of the two data sets. If $\log(P_{err}) > THRESHOLD$ (=constant), detection of the watermark is assumed to be successful. The function returns also the ratio of the correctly detected pixels per total number of watermarked pixels.

RETURN VALUES: 0, 1, -1, -9, -21

SEE ALSO : add_wtr_watermark().

References

- [PIT95] I. Pitas, T. Kaskalis, "Applying signatures on digital images", IEEE Workshop on Non-linear Signal and image Processing, June 1995, pp 460-463.
- [NIK96] , N. Nikolaidis, I. Pitas, "Copyright Protection of Images Using Robust Digital Signatures", IEEE Int. Conference on Acoustics, Speech and Signal Processing, June 1996.

Index

`add_wtr_watermark`, 3

`detect_wtr_watermark`, 4

Embedding level, 2

Image watermarking, 1

Installation, 1